PENGEMBANGAN APLIKASI MANAJEMEN *EVENT* BERBASIS WEB (Studi Kasus: Fakultas Ilmu Administrasi Universitas Brawijaya Malang)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hendro Febrian Bachri
NIM: 145150200111151

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN APLIKASI MANAJEMEN *EVENT* BERBASIS WEB (Studi Kasus: Fakultas Ilmu Administrasi Universitas Brawijaya Malang)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Hendro Febrian Bachri
NIM: 145150200111151

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Priyambadha, S.Kom., M.Kom	Denny Sagita Rusdianto, S.Kom., M.Kom
NIP: 19820909 200812 1 004	NIP: 19851124 201504 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018

Hendro Febrian Bachri

NIM: 145150200111151

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan Rahmat, Taufik dan Hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Pengembangan Aplikasi Manajemen *Event* Berbasis Web (Studi Kasus: Fakultas Ilmu Administrasi Universitas Brawijaya Malang)". Shalawat dan salam juga tidak lupa penulis limpahkan kepada Nabi Muhammad SAW.

Tujuan dari penyusunan skripsi ini adalah sebagai salah satu persyaratan untuk memperoleh gelar Sarjana pada Program Studi Teknik Informatika di Fakultas Ilmu Komputer, Universitas Brawijaya Malang.

Didalam pengerjaan skripsi ini, penulis mendapatkan banyak bantuan baik secara moril maupun secara materiil. Oleh sebab itu, dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sedalam - dalamnya kepada:

1. Allah SWT atas segala limpahan Rahmat, Taufik dan Hidayah-Nya.
2. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
4. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya Malang.
5. Bapak Bayu Priyambadha, S.Kom., M.Kom selaku pembimbing skripsi pertama yang telah banyak membantu, membimbing dan mengarahkan penulis dalam pengerjaan skripsi ini.
6. Bapak Denny Sagita Rusdianto, S.Kom., M.Kom selaku pembimbing skripsi kedua yang banyak memberikan arahan dan berbagai ide dalam pengerjaan skripsi ini.
7. Bapak Rizki Yudhi Dewantara, S.Sos., M.PA dan Bapak Swasta Priambada, S.Sos., M.AB serta Aria Adi Negoro yang telah membantu penulis dalam pengumpulan data pada penelitian ini.
8. Kedua orang tua atas jasa-jasa, doa, kesabaran yang tidak pernah lelah dalam mendidik penulis semenjak kecil.
9. Teman-teman OTW Kewong, Viktor Kekexelen, Tukang Left serta teman - teman lain yang tidak bisa penulis sebutkan satu-persatu.

Semoga Allah SWT memberikan balasan yang berlipat ganda kepada semuanya. Penulis menyadari didalam penyusunan skripsi ini, masih sangat banyak kekurangan baik dari segi format laporan maupun isinya. Maka dari itu, penulis sangat mengharapkan kritik dan saran yang membangun demi perbaikan selanjutnya. Semoga skripsi ini dapat bermanfaat bagi pembaca dan penulis.

Malang, 2 Agustus 2018

Penulis

hendrofebrian43@gmail.com

ABSTRAK

Event dapat didefinisikan sebagai suatu kegiatan yang diselenggarakan guna memperingati hal-hal penting sepanjang hidup manusia baik secara individu ataupun kelompok (Noor, 2013). Fakultas Ilmu Administrasi Universitas Brawijaya (FIA UB) sangat aktif dalam menyelenggarakan *event* seperti seminar, *workshop*, lomba dan lain-lain. *Event* tersebut dapat diselenggarakan oleh organisasi mahasiswa atau unit kerja fakultas. Untuk dapat mengadakan *event* di FIA UB tentunya pengaju *event* harus melalui prosedur tertentu. Prosedur pengajuan *event* berbeda-beda tergantung pengaju event, *resource* yang diperlukan serta lokasi penyelenggaraan *event*. Namun terdapat beberapa permasalahan yang timbul dalam proses pengajuan *event* di FIA UB. Dari perspektif pengaju *event*, masalah yang timbul antara lain pengaju *event* tidak mengetahui status pengajuan *event* yang diajukannya, tidak mengetahui status ketersediaan *venue* dan banyak pengaju *event* yang tidak mengetahui alur pengajuan *event*. Dari perspektif TU Subbag Umum & Perlengkapan, pengecekan ketersediaan *venue* dan pencatatan data peminjaman *venue* yang masih dilakukan didalam *file* kurang efisien dan memakan waktu. Sehingga dibuatlah aplikasi Manajemen *Event* untuk mempermudah proses pengajuan *event*, peminjaman *venue* dan pencairan dana. Untuk mengimplementasikan aplikasi Manajemen *Event*, penulis akan menggunakan Node.js dengan bantuan *framework* Express sebagai *back-end* dan SemanticUI yang merupakan *framework* CSS untuk membantu mempercepat pengimplementasian *front-end* dari aplikasi ini. Pengujian dilakukan dengan menggunakan pengujian unit, pengujian validasi dan pengujian *browser compatibility*. Pada pengujian *unit,* seluruh *independent paths* telah diuji dan seluruhnya *valid*. Hasil pengujian validasi juga menunjukkan nilai 100% *valid*. Sedangkan hasil pengujian *browser compatibility* menunjukkan aplikasi dapat dijalankan secara sempurna (tanpa kehilangan fungsionalitas) di peramban Edge, Safari, Chrome, Firefox dan Opera.

Kata kunci: Manajemen *Event*, Node.js, Express.js, SemanticUI

ABSTRACT

Events can be defined as activities that are held to celebrate the important things throughout human life both individually and in groups (Noor, 2013). Faculty of Administrative Sciences, University of Brawijaya (FIA UB) is really active in organizing events such as seminars, workshops, competitions and others. Those events can be held by student organization or faculty's internal unit. There is an event proposal submission procedure for holding an event in FIA UB. The event proposal submission procedure is different and it depends on who is the event organizer, what resources are required and the where the event will be held. There are some problems that arise in the process of event proposal submission in FIA UB. From the event organizers perspective, they don't know the status of the submission of the event proposals that they have submitted, they also don't know the status of the venue's availability and many of event organizer do not know the flow of the event submission. From perspective of the General and Equipment Subdivision, the checking of venue's availability and recording of venue's reservation that are still performed manually in files is less efficient and time-consuming. So, the author have a solution to make an application called Event Management application. This Event Management application will be implemented using Node.js with the Express.js framework as a back-end and SemanticUI which is a CSS framework to help speed up the implementation of the front-end of this application. The testing conducted using unit tesing, validation testing and browser compatibility testing methods. In unit testing, all of independent paths have been tested and all of them are valid. The validation testing also shows result 100% valid. While browser compatibility testing shows that this application can be perfectly (without missing functionality) in Edge, Safari, Chrome, Firefox and Opera.

Keywords: *Event Management*, *Node.js, Express.js, SemanticUI*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Event dapat didefinisikan sebagai suatu kegiatan yang diselenggarakan guna memperingati hal-hal penting sepanjang hidup manusia baik secara individu ataupun kelompok dan terikat secara adat, budaya, tradisi dan agama. *Event* diselenggarakan dengan tujuan dan waktu yang spesifik serta melibatkan kelompok masyarakat tertentu (Noor, 2013). Fakultas Ilmu Administrasi Universitas Brawijaya (FIA UB) sangat aktif dalam menyelenggarakan *event-event* seperti seminar, *workshop*, lomba dan lain-lain. *Event-event* tersebut dapat diselenggarakan oleh organisasi mahasiswa, dekanat, unit kerja fakultas ataupun peminjam *venue* dari luar fakultas. Untuk menyelenggarakan *event* di FIA UB, tentunya harus melewati suatu prosedur. Alur pengajuan *event* berbeda-beda tergantung pengaju *event* (dekanat, unit internal fakultas atau organisasi mahasiswa), *resource* yang diperlukan (keuangan dan *venue*) serta lokasi penyelenggaraan *event* (didalam kota atau diluar kota).

Terdapat beberapa permasalahan yang timbul dalam proses pengajuan *event* di Fakultas Ilmu Administrasi UB. Dari perspektif pengaju *event*, masalah yang timbul antara lain pengaju event tidak mengetahui status pengajuan *event* yang diajukannya dan masih banyak pengaju *event* khususnya dari organisasi mahasiswa yang tidak mengetahui prosedur pengajuan *event*. Selain itu, pengaju *event* mahasiswa juga tidak langsung mendapatkan notifikasi apabila *event* yang diajukannya ditunda atau dipindah tempatkan dikarenakan *venue* yang sudah direservasi ternyata akan dipakai oleh *event* unit internal fakultas yang lebih diprioritaskan. Banyaknya jumlah *event* dan jumlah *venue* yang terbatas juga menjadi masalah tersendiri bagi para pengaju event dalam hal pemilihan *venue*. Hal ini dikarenakan, pengaju *event* tidak mengetahui status ketersediaan *venue* secara keseluruhan. Pada Lampiran L dapat dilihat bahwa jumlah *venue* yang biasa digunakan untuk menyelenggarakan event ada 8 *venue* (tidak termasuk ruang kelas) sedangkan jumlah *event* yang diselenggarakan di FIA UB per bulannya adalah 95 *event* (Agustus 2017).

Dari perspektif TU Subbag Umum & Perlengkapan, pengecekan ketersediaan *venue* dan pencatatan data peminjaman *venue* yang masih dilakukan didalam *file* kurang efisien dan memakan waktu terlebih jika *event* tersebut merupakan *event* rutin (event yang dilaksanakan hampir setiap hari atau mingguan), yang membuat staff TU Subbag Umum & Perlengkapan harus mencatat data-data peminjaman *venue* yang sama secara berulang-ulang. Selain itu pengarsipan proposal *event* juga menjadi kendala tersendiri dikarenakan FIA UB belum memiliki ruang arsip. Masalah lain yang timbul adalah jika pihak-pihak yang berperan dalam menyetujui pengajuan *event* dalam hal ini Dekanat, Kepala Jurusan atau Kepala Tata Usaha tidak ada di tempat dikarenakan sedang bertugas keluar kota atau karena alasan lain. Hal ini tentunya akan menghambat proses pengajuan event karena proses persetujuan terhadap proposal *event* tidak dapat dilakukan secara *remote*.

Sehingga, untuk mempermudah proses manajemen *event* dan untuk membantu mengatasi kekurangan-kekurangan yang timbul akibat prosedur manual pengajuan *event* di FIA UB, penulis memiliki solusi yaitu dengan membangun suatu aplikasi untuk mempermudah proses pengajuan event, peminjaman *venue* dan pencairan dana yang disebut aplikasi Manajemen *Event*. Aplikasi ini akan dikembangkan menjadi aplikasi berbasis web. Hal ini dikarenakan selain permintaan dari *customer*, aplikasi berbasis web memiliki banyak kelebihan diantaranya memiliki kompatibilitas dan availabilitas yang tinggi dan tidak memerlukan instalasi dalam penggunaannya sehingga dapat langsung digunakan.

Pada penelitian yang berjudul "*Is Node.js a viable option for building modern web applications? A performance evaluation study*" oleh (Chaniotis et. al., 2014) diperoleh bahwa Node.js secara mutlak mengalahkan PHP dalam performa komputasi karena Node.js jauh lebih efisien dalam hal manajemen *memory* dan pemanfaatan *processing power*. Kesimpulan dari paper ini adalah penulis sangat merekomendasikan untuk menggunakan *end-to-end* Javascript sebagai pilihan yang dapat digunakan untuk membangun aplikasi *website* modern. Namun, menurut penelitian yang berjudul "*Assessing the Security of Node.js Platform*", diperoleh analisis bahwa penggunaan Node.js harus dihindari ketika pembuatan aplikasi yang memiliki kebutuhan yang ketat akan keamanan. Alasannya adalah *platform* tersebut belum matang karena merupakan *platform* yang relatif baru sehingga masih memiliki celah-celah keamanan (Ojamaa & Duuna, 2012).

Sehingga berdasarkan dua penelitian diatas, untuk mengimplementasikan aplikasi ini penulis akan menggunakan Node.js sebagai *back-end* dan SemanticUI yang merupakan *framework* CSS untuk membantu mempercepat pengimplementasian *front-end* dari aplikasi ini. Paradigma pemrograman yang akan digunakan penulis pada penelitian ini adalah *Object Oriented*. Untuk itu, penulis akan menggunakan salah satu *framework* Node.js yang menggunakan paradigma pemrograman *Object Oriented* yaitu Express.js. Express.js merupakan salah satu *framework* Node.js yang paling populer, banyak digunakan, *well-documented* dan dinilai paling matang saat ini (Mayven, 2017). Pada penelitian ini, penulis juga akan menggunakan aplikasi Google Calendar untuk menampilkan kalender *event*. Alasan penulis menggunakan Google Calendar adalah selain untuk mempercepat proses pengembangan aplikasi manajemen *event* juga untuk mempermudah pengguna jika hanya ingin melihat kalender *event*. Google Calendar sangat mudah disinkronisasi dengan aplikasi seperti Samsung Calendar, iCal atau aplikasi lain yang sejenis sehingga kalender *event* dapat diakses kapan pun walaupun pengguna tidak sedang terkoneksi dengan internet.

Karena seluruh kebutuhan dari aplikasi Manajemen *Event* sudah diketahui di awal, maka penulis akan menggunakan *Software Development Life Cycle* (SDLC) *Waterfall.* Hal ini sesuai dengan penelitian yang berjudul "*A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*" yang berkesimpulan bahwa *Software Development Life Cycle* (SDLC)

Waterfall dapat digunakan ketika kebutuhan-kebutuhan dari suatu sistem sudah diketahui diawal, *clear* dan *fixed* (Alshamrani & Bahattab, 2015).

Aplikasi Manajemen *Event* diharapkan dapat membuat proses pengajuan event di Fakultas Ilmu Administrasi menjadi lebih mudah dan efisien. Selain itu, pengimplementasian *framework* Express.js diharapkan akan mempercepat proses pengembangan aplikasi ini.

1.2 Rumusan Masalah

Masalah yang akan diteliti adalah sebagai berikut:

1. Bagaimana hasil analisis dan elisitasi kebutuhan sistem dari prosedur pengajuan *event* yang berlaku di Fakultas Ilmu Administrasi Universitas Brawijaya?
2. Bagaimana hasil perancangan, implementasi dan pengujian aplikasi Manajemen *Event* yang dibangun menggunakan SDLC *Waterfall* dan menggunakan *framework* Express.js?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

a. Untuk membangun aplikasi Manajemen *Event* yang dapat digunakan untuk membantu proses pengajuan *event*, peminjaman *venue* dan pengajuan dana di Fakultas Ilmu Administrasi Universitas Brawijaya Malang.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah:

a. Membantu proses pengajuan *event*, mempermudah pengarsipan proposal *event*, peminjaman *venue,* pengajuan dana serta mempermudah mendapatkan informasi *event* di Fakultas Ilmu Administrasi Universitas Brawijaya.

1.5 Batasan Masalah

Batasan penelitian ini adalah:

a. Penelitian ini hanya membahas proses analisis dan elisitasi kebutuhan, perancangan, pengimplementasian dan pengujian aplikasi Manajemen *Event* di Fakultas Ilmu Administrasi Universitas Brawijaya yang dibangun menggunakan SDLC *Waterfall* dan menggunakan *framework* Express.js.

1.6 Sistematika Pembahasan

Sistematika pada penulisan penelitian ini adalah sebagai berikut:

1. BAB I PENDAHULUAN
   Bab ini akan berisi latar belakang, identifikasi masalah, rumusan masalah penelitian, tujuan dan manfaat penelitian, batasan penelitian, sistematika penelitian serta jadwal penelitian.

2. BAB II LANDASAN KEPUSTAKAAN
   Pada bagian ini akan dikaji teori-teori tentang permasalahan yang diangkat, juga akan diuraikan mengenai solusi untuk menyelesaikan permasalahan tersebut.
3. BAB III METODOLOGI PENELITIAN
   Bagian ini akan membahas langkah-langkah yang akan dilakukan untuk menyelesaikan permasalahan yang diangkat di penelitian ini.
4. BAB IV ANALISIS KEBUTUHAN
   Bagian ini akan membahas terkait elisitasi dan analisis kebutuhan sistem.
5. BAB V PERANCANGAN DAN IMPLEMENTASI
   Bab ini akan membahas perancangan sistem yang akan digunakan sebagai dasar proses implementasi yaitu mengubah perancangan sistem kedalam kode program.
6. BAB VI PENGUJIAN
   Pada bab ini akan dibahas tentang hasil pengujian dari perangkat lunak yang telah dibuat.
7. BAB VII KESIMPULAN DAN SARAN
   Pada bab ini akan dipaparkan kesimpulan dari penelitian serta saran untuk pengembangan penelitian.

1.7 Jadwal Penelitian

Jadwal penelitian diilustrasikan pada Tabel 1.1 dibawah ini:

Tabel 1.1 Jadwal Penelitian

No	Jenis Kegiatan	Bulan			
		1	2	3	4
1	Studi literatur				
2	Analisis dan elisitasi kebutuhan				
3	Perancangan				
4	Implementasi				
5	Pengujian				
6	Penarikan kesimpulan				

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang berisi kajian teoritis dan kajian empiris yang diperlukan didalam penelitian. Kajian teoritis akan membahas tentang teori yang diperlukan dalam penelitian. Sedangkan kajian empiris membahas tentang penelitian yang telah ada sebelumnya atau informasi yang diperoleh telah dibuktikan dengan penelitian/observasi.

2.1 Kajian Pustaka

Perbandingan performa Node.js, Python web dan PHP telah dilakukan di penelitian yang berjudul "*Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js*" oleh (Lei, et. al., 2014) diperoleh kesimpulan performa Node.js jauh lebih baik daripada PHP didalam situasi *high concurrency*. Python-web juga tidak sebaik Node.js dalam mengkomputasi proses yang *intensive* dalam aplikasi web.

Di penelitian lain yang berjudul "*Is Node.js a viable option for building modern web applications? A performance evaluation study*" oleh (Chaniotis, et. al., 2014) diperoleh bahwa Node.js secara mutlak mengalahkan PHP dalam performa komputasi karena Node.js jauh lebih efisien dalam manajemen *memory*. Kesimpulan dari *paper* ini adalah penulis sangat merekomendasikan untuk menggunakan *end-to-end* Javascript sebagai pilihan yang dapat digunakan untuk membangun aplikasi *website modern*.

Namun, menurut penelitian yang berjudul "*Assessing the Security of Node.js Platform*", diperoleh analisis bahwa penggunaan Node.js harus dihindari ketika pembuatan aplikasi yang memiliki kebutuhan yang ketat akan keamanan. Hal ini dikarenakan Node.js merupakan *platform* yang belum matang karena merupakan *platform* yang relatif baru sehingga masih memiliki celah-celah keamanan (Ojamaa & Duuna, 2012).

Karena seluruh kebutuhan dari aplikasi Manajemen *Event* sudah diketahui di awal, maka penulis akan menggunakan *Software Development Life Cycle* (SDLC) *Waterfall.* Hal ini sesuai dengan penelitian yang berjudul *"A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*" yang berkesimpulan bahwa *Software Development Life Cycle* (SDLC) *Waterfall* dapat digunakan ketika kebutuhan-kebutuhan dari aplikasi Manajemen *Event* sudah diketahui diawal, *clear* dan *fixed* (Alshamrani & Bahattab, 2015).

2.2 Event

Event dapat didefinisikan sebagai suatu kegiatan yang diselenggarakan guna memperingati hal-hal penting sepanjang hidup manusia baik secara individu ataupun kelompok dan terikat secara adat, budaya, tradisi dan agama. *Event* diselenggarakan dengan tujuan dan waktu yang spesifik serta melibatkan kelompok masyarakat tertentu (Noor, 2013). Menurut Any Noor (2013), berdasarkan ukurannya dapat dikategorikan menjadi:

a. *Mega Event*

   Merupakan suatu *event* yang sangat besar dan dapat memberikan pengaruh terhadap ekonomi pada masyarakat bahkan negara.

b. *Hallmark Event*

   Merupakan *event* yang identik terhadap karakter dari suatu daerah.

c. *Major Event*

   Misalnya *event* olahraga yang diselenggarakan rutin setiap tahun (*annual*).

Any Noor (2013) membagi *event* berdasarkan kategori *special event* yaitu:

a. *Leisure Event*

   Misalnya olimpiade, *world cup* dan *asian games.*

b. *Cultural Event*

   Merupakan *event* yang berkaitan dengan kegiatan kebudayaan.

c. *Personal Event*

   Merupakan *event* yang hanya melibatkan orang-orang dilingkungan sekitar misalnya keluarga dan teman.

d. *Organizational Event*

   Merupakan *event* yang diadakan oleh suatu organisasi atau instansi dan bertujuan untuk meningkatkan popularitas dari organisasi atau instansi tersebut.

2.3 Proses-proses dalam Rekayasa Perangkat Lunak

Definisi rekayasa perangkat lunak (*software engineering*) menurut IEEE [IEE93a] dikutip dari Pressman (2015), Rekayasa perangkat lunak adalah pengaplikasian atau pendekatan yang sistematis, berdisiplin, pendekatan kuantitatif dalam proses pengembangan, pengoperasian dan pemeliharaan (*maintenance*) suatu perangkat lunak.

Proses rekayasa perangkat lunak adalah suatu rangkaian kegiatan yang saling terkait yang bertujuan untuk mengembangkan suatu perangkat lunak. Secara garis besar, rekayasa perangkat lunak terdiri dari 7 tahap yaitu analisis dan elisitasi kebutuhan, perancangan, pengkodean, pengujian, *deployment* dan *maintenance*.

2.3.1 Elisitasi dan Analisis Kebutuhan

Elisitasi atau penggalian kebutuhan adalah proses mengumpulkan informasi tentang sistem yang akan dibuat, sistem yang telah ada sekarang ini lalu menentukan calon aktor dan kebutuhan sistem dari informasi ini. Pengembang berinteraksi dengan *stakeholder* melalui *interview* dan observasi (Sommerville, 2011).

Analisis kebutuhan merupakan proses untuk menganalisis hasil dari tahap elisitasi kebutuhan. Hasil dari proses analisis tersebut adalah calon aktor sistem serta definisi dan spesifikasi kebutuhan sistem. Definisi dan spesifikasi kebutuhan yang dihasilkan haruslah sedetail mungkin dan bebas dari ambiguitas.

2.3.2 Perancangan

Perancangan merupakan proses membangun model abstrak dari suatu sistem. Masing-masing model mempresentasikan perspektif yang berbeda dari suatu sistem. Perancangan sistem bertujuan untuk merepresentasikan sistem menggunakan notasi grafis yang sekarang ini berbasis pada notasi di *Unified Modelling Language* (Sommerville, 2011).

Menurut survey pada tahun 2007 oleh Erickson dan Siau dikutip dari (Sommerville, 2011), menunjukan bahwa kebanyakan pengguna diagram UML (*Unified Modelling Language*) berpendapat bahwa lima diagram UML mampu merepresentasikan hal yang esensial dari suatu sistem. Kelima diagram tersebut adalah:

a. *Activity diagram* yang menunjukkan aktivitas yang terlibat dalam suatu proses atau dalam suatu pemrosesan data.
b. *Use case diagram* menunjukkan interaksi antara sistem dan lingkungannya.
c. *Sequence diagram* menunjukkan interaksi antara aktor dan sistem serta antara komponen sistem.
d. *Class diagram* menunjukkan kelas-kelas yang ada di dalam sistem dan relasi antar kelas-kelas tersebut.
e. *State diagram* menunjukkan bagaimana sistem bereaksi terhadap *events* internal dan eksternal.

2.3.3 Pengkodean

Pengkodean merupakan proses interpretasi diagram-diagram yang telah dibuat pada fase perancangan menjadi kode program. Pengkodean adalah proses untuk menterjemahkan perancangan kedalam baris-baris kode sesuai bahasa pemrograman yang digunakan. Dalam pengkodean, programmer harus mengikuti setiap detail dari rancangan yang telah dibuat sebelumnya.

2.3.4 Pengujian

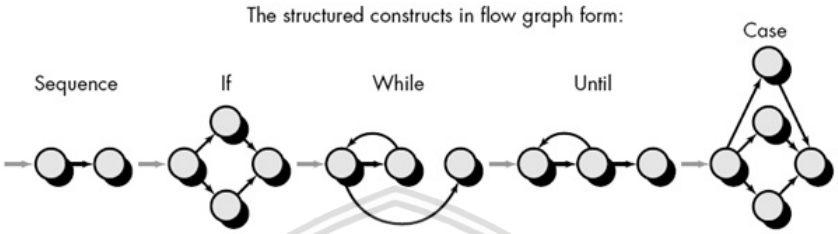
Pengujian adalah serangkaian kegiatan yang dapat direncanakan terlebih dahulu dan dilakukan secara sistematis (Pressman, 2015). Software diuji untuk menemukan kesalahan dari perangkat lunak yang diakibatkan kesalahan saat fase desain dan atau fase pengkodean. Didalam SDLC (*Software Development Life Cycle*), pengujian dapat dimulai dari fase pengumpulan kebutuhan dan dilanjutkan sampai tahap *deployment* dari suatu perangkat lunak. Namun hal ini juga tergantung kepada SDLC yang digunakan. Pengujian perangkat lunak berdasarkan jenis kebutuhan sistem yang diuji dapat dibagi menjadi 2 yaitu

pengujian fungsional dan pengujian non-fungsional yang akan dijelaskan berikut ini.

a. Pengujian Fungsional

Terdapat 2 level pengujian kebutuhan fungsional yang akan dijelaskan berikut ini.

1. Pengujian Unit



Gambar 2.1 Notasi Struktur *Control Flow* pada *Flow Graph*

Sumber: (Meiliana, 2017)

Pengujian unit adalah proses untuk menguji komponen program seperti *method* atau objek dari suatu kelas (Sommerville, 2011). Pada penelitian ini, penulis akan menggunakan *White Box Testing*. Metode yang akan digunakan adalah *Basis Path Testing. Basis Path Testing* adalah salah satu metode yang digunakan dalam pengujian unit dengan menggunakan *flow graph*. *Flow graph* merupakan suatu *graph* yang digunakan untuk mengilustrasikan aliran kontrol didalam suatu *method*. *Flow graph* direpresentasikan dengan *node* dan *edge*. *Node* mewakili *decisions* dan *edge* mewakili aliran kontrol. Gambar 2.1 diatas menunjukkan notasi struktur kontrol pada *flow graph.*

Flow graph dapat digunakan untuk mengidentifikasi *independent path* yang ada. *Flow graph* juga dapat digunakan untuk menghitung *cyclomatic complexity* atau V(G) dari suatu *method. Independent path* adalah setiap jalur yang ada di *flow graph,* dimana setiap *node* minimal harus dilewati sebanyak satu kali yang memastikan bahwa seluruh *case* yang mungkin ada pada suatu *method* telah dijalankan dan diuji minimal satu kali. *Cyclomatic compexity* adalah suatu *software metric* yang digunakan untuk menunjukkan ukuran kuantitatif dari kompleksitas logika pada suatu *method* (Meiliana, 2017). *Cyclomatic complexity* atau V(G) dapat dihitung menggunakan rumus:

a. V(G) = *edges* – *nodes* +2
b. V(G) = *region*
c. V(G) = *desicion* (*predicate nodes*) +1

2. Pengujian Validasi/Sistem

Pengujian validasi atau pengujian sistem dilakukan dengan membuat daftar *test case.* Kemudian tiap *test case* akan diuji untuk mengetahui apakah sistem

menampilkan hasil yang sesuai dengan *expected result*. Jika sesuai maka pengujian validasi untuk *test case* tersebut dianggap *valid*.

b. Pengujian Non Fungsional

1. *Browser Compatibility Testing*

Browser Compatibility Testing merupakan salah satu jenis pengujian non-fungsional yang bertujuan untuk menguji apakah seluruh fitur yang ada pada suatu aplikasi berbasis web dapat dijalankan di berbagai web *browser*. Pada penelitian ini, penulis akan menggunakan *tools* bernama SortSite.

SortSite merupakan aplikasi berbasis desktop yang dapat digunakan untuk menguji kebutuhan non fungsional dari suatu sistem yaitu *accessibility*, *privacy*, *usability* dan *compatibility*. SortSite bekerja dengan cara memeriksa beberapa hal berikut ini:

a. Tag HTML yang tidak didukung oleh beberapa *browser*
b. Fitur CSS yang tidak didukung oleh beberapa *browser*
c. *Vendor specific* HTML dan Javascript
d. Format gambar yang tidak didukung beberapa *browser*
e. Teknologi yang tidak didukung oleh beberapa *browser*

2.3.5 *Deployment*

Deployment adalah proses untuk menyiapkan perangkat agar untuk siap dipasarkan atau digunakan oleh pelanggan. Elemen-elemen utama yang harus disertakan saat proses *deployment* diantaranya:

a. *Executable file*: merupakan kode program yang telah melalui proses pengujian sehingga minim *bug*.
b. Dokumentasi: dokumen yang berisi informasi detail terkait perangkat lunak mulai dari analisis kebutuhan sampai pengujian perangkat lunak.

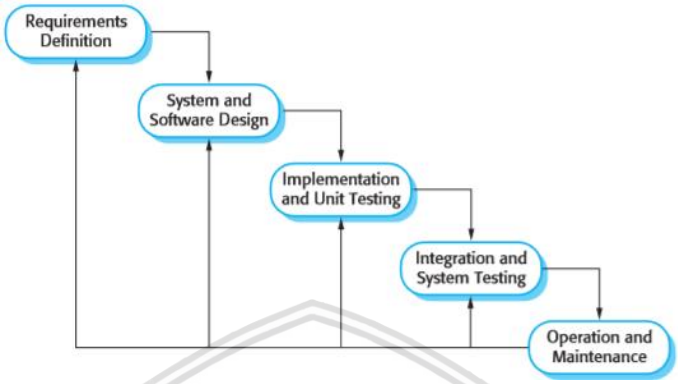
2.3.6 Maintenance

Software maintenance adalah proses *general* dari mengubah suatu sistem yang dilakukan setelah sistem telah diantarkan ke pengguna. Ada 3 tipe *software maintenace* menurut (Sommerville, 2011) yaitu:

1. *Fault repairs,* terdiri dari *coding errors, design errors* dan *requirements error.*
2. *Environmental adaptation,* merupakan tipe dari *maintenance* yang diperlukan apabila ada aspek dari lingkungan sistem yang berubah seperti *hardware* dan *platform* OS.
3. *Functionality addition,* merupakan tipe dari *maintenance* yang dibutuhkan ketika ada kebutuhan yang berubah dikarenakan proses bisnis yang berubah.

2.4 Software Development Life Cycle (SDLC) *Waterfall*

Model *waterfall* adalah model proses pertama yang dikenalkan. Waterfall model disebut juga *linear-sequential life cycle model.* Model *waterfall* merupakan SDLC yang sangat mudah untuk dipahami dan digunakan.



Gambar 2.2 *Waterfall Model*

Sumber: (Sommerville, 2011)

Gambar 2.2 merupakan ilustrasi dari SDLC *Waterfall.* Penjelasan detail mengenai Gambar 2.2 adalah sebagai berikut:

a. Pada tahap pertama yaitu *Requirement Definition*, seluruh kebutuhan harus sudah didefinisikan dan dispesifikasikan secara lengkap, *clear* dan *fixed.*

b. Pada tahap kedua yaitu *System and Software Design*, definisi dan spesifikasi kebutuhan yang dihasilkan pada tahap pertama akan dipelajari untuk kemudian akan dibuat perancangan sistemnya. Perancangan berfungsi untuk membantu mendefiniskan arsitektur sistem secara keseluruhan.

c. Pada tahap ketika akan dilakukan implementasi atau interpretasi perancangan kedalam kode program menggunakan bahasa pemrograman yang tertentu. Selain itu akan dilakukan pengujian unit untuk menguji *method-method* yang ada dalam sistem.

d. Tahap keempat merupakan pengujian integrasi dan pengujian validasi.

e. Tahap yang terakhir adalah *maintenance* atau pemeliharaan perangkat lunak. *Operation* dan *Maintenance* merupakan fase terpanjang. *Maintenance* termasuk memperbaiki *error,* memperbaiki implementasi dan meningkatkan servis dari suatu sistem ketika suatu kebutuhan baru ditemukan.

2.5 Pemodelan Kebutuhan dan Perancangan Sistem dengan Pendekatan Berorientasi Objek

Untuk memodelkan sistem yang dibangun dengan pemrograman dengan pendekatan berorientasi objek, penulis akan menggunakan diagram beberapa

diagram UML yaitu *Use Case Diagram, Sequence Diagram* dan *Class Diagram*. Kemudian penulis juga menggunakan State Transition Diagram (STD) untuk menggambarkan alur atau status disposisi proposal atau persetujuan *event.*

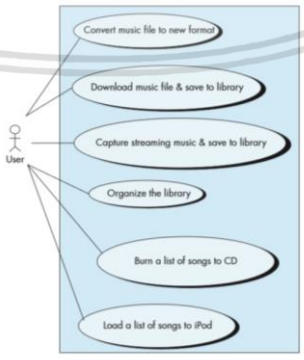
2.5.1 UML (Unified Modelling Language) Diagram

Unified Modelling Language (UML) diagram adalah bahasa standar yang digunakan untuk memvisualisasikan rancangan perangkat lunak. UML dapat digunakan untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan suatu perangkat lunak. *Designer* perangkat lunak akan membuat UML diagram untuk membantu pengembang perangkat lunak dalam membangun *software* tersebut (Pressman, 2015). Untuk pemrograman berorientasi objek, ada beberapa jenis diagram UML yang sering digunakan yaitu *class diagram, use case diagram* dan *sequence diagram*.

2.5.1.1 Use Case Diagram

Use case diagram membantu pengembang aplikasi dalam proses menentukan fungsionalitas dan fitur dari *software* dari sudut pandang pengguna. Sebuah *use case* menggambarkan bagaimana pengguna berinteraksi dengan sistem dengan cara mendefinisikan langkah-langkah yang diminta untuk menyelesaikan suatu tujuan yang spesifik (Pressman, 2015).

Untuk mendetailkan suatu *use case* dapat dilakukan dengan menggunakan *use case scenario. Use case scenario* berisi aktor yang dapat mengakses use case, tujuan *use case*, prekondisi, *main flow*, *alternative flow* dan *post* kondisi. Pre kondisi merupakan kondisi yang harus terpenuhi sebelum *use case* dijalankan. *Main flow* merupakan alur utama atau alur dasar dari *use case* dimana dalam suatu *use case* hanya boleh memiliki satu *main flow*. *Alternative flow* berisi alur-alur selain alur dasar misalkan kondisi-kondisi yang menyebabkan alur dasar tidak dapat dijalankan. *Post* kondisi merupakan kondisi yang tercapai setelah suatu *use case* dijalankan. Gambar 2.3 dibawah ini merupakan contoh *Use Case Diagram*.

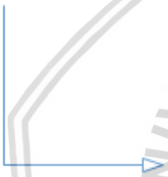
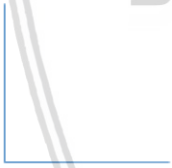


Gambar 2.3 Struktur *Use Case Diagram*

Sumber: (Pressman, 2015)

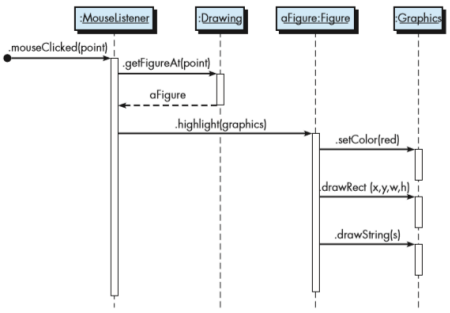
Tabel 2.1 berikut ini merupakan notasi-notasi yang paling sering digunakan *use case diagram*.

Tabel 2.1 Notasi di Use Case Diagram

No	Gambar	Nama	Keterangan
1		*Aktor*	Merepresentasikan pengguna aplikasi.
2		*Use Case*	Mendeskripsikan aksi-aksi yang dapat dilakukan sistem.
3		*Include*	Hubungan yang berarti *use case* target membutuhkan *use case* sumber untuk menjalankan fungsinya
4		*Extend*	Hubungan yang berarti *use case* target memperluas/mengextend perilaku
5		Generalisasi	*Ancestor* berbagi perilaku dengan *descendant*.
6		Asosiasi	Menghubungkan objek satu dengan objek lainnya

2.5.1.2 Sequence Diagram

Sebuah *sequence diagram* digunakan untuk menunjukkan komunikasi yang dinamis antar *object* selama proses eksekusi berlangsung (Pressman, 2015). Gambar 2.4 dibawah ini merupakan contoh dari *Sequence Diagram.*



Gambar 2.4 Struktur *Sequence Diagram*

Sumber: (Pressman, 2015)

Tabel 2.2 dibawah ini menunjukkan elemen-elemen utama yang ada di *sequence diagram*.

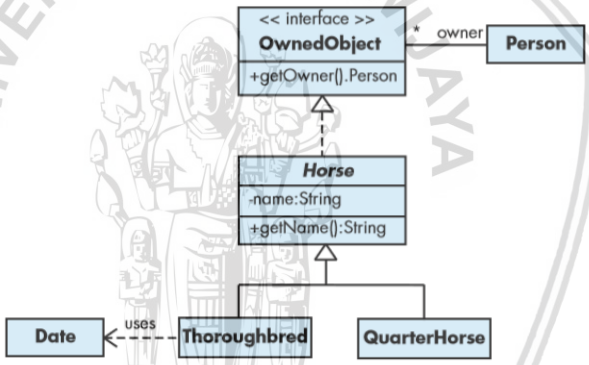
Tabel 2.2 Notasi di Sequence Diagram

No	Gambar	Nama	Keterangan
1		*Activation Bar*	Merepresentasikan waktu hidup dari suatu objek
2		*Aktor*	Merepresentasikan pengguna aplikasi
3		*Lifeline*	Tempat untuk meletakkan *activation bar*
4		*Boundary Lifeline*	Merepresentasikan antarmuka sistem
5		*Control Lifeline*	Merepresentasikan objek yang memiliki tugas sebagai pengatur sistem (*controller*)
6		*Entity Lifeline*	Merepresentasikan objek yang memiliki tugas yang berkaitan dengan data sistem (model)
7		*Alternative Fragment*	Digunakan apabila terdapat *alternative flow*
8		*Loop Fragment*	Digunakan apabila terdapat perulangan

9		*Self Message*	Dapat digunakan untuk me*return* pesan ke objek itu sendiri
10		*Return Message*	Digunakan untuk me*return* pesan ke objek lain
11		*Message*	Digunakan untuk mengirimkan pesan ke objek lain

2.5.1.3 Class Diagram

Class Diagram digunakan untuk merepresentasikan struktur kelas dan relasinya beserta keseluruhan atribut dan *method* yang ada didalamnya. Kelas diagram menyediakan struktur statis dari suatu sistem (Pressman, 2015). Gambar 2.5 dibawah ini merupakan ilustrasi dari kelas diagram.

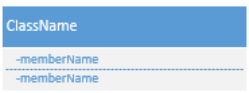


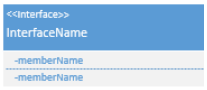
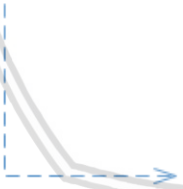
Gambar 2.5 Struktur Kelas Diagram

Sumber: (Pressman, 2015)

Tabel 2.3 dibawah ini merupakan elemen-elemen utama dari kelas diagram.

Tabel 2.3 Notasi di *Class* Diagram

No	Gambar	Nama	Keterangan
1		Kelas	Merepresentasikan struktur kelas
2		*Package*	Untuk mengelompokkan kelas-kelas (merepresentasikan *package*)

3		*Note*	Digunakan untuk membuat catatan
4		*Interface*	Digunakan untuk mewakili kelas *interface*
5		Generalisasi	Digunakan untuk menggambarkan hubungan generalisasi
6		Asosiasi	Untuk mendefinisikan hubungan asosiasi
7		*Interface Realization*	Menjelaskan hubungan antara *interface* dengan realisasi *interface*.
8		Dependensi	Mendefinisikan ketergantungan/dependensi antar kelas
9		Agregasi	Mendefinisikan bahwa satu kelas dengan kelas lain terikat cukup kuat (*has a)*
10		Komposisi	Mendefinisikan bahwa satu kelas dengan kelas lain terikat sangat kuat (*is a/is part*)

11		Asosiasi berarah	Merepresentasikan hubungan asosiasi yang searah

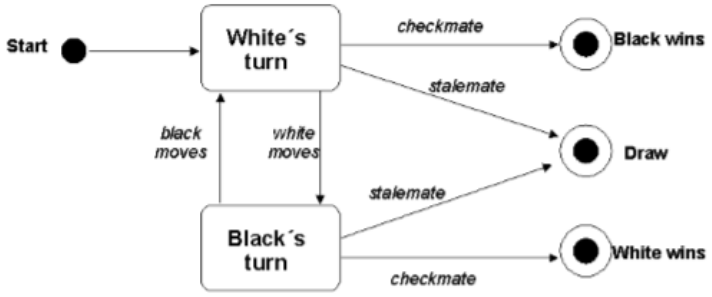
2.5.1.4 State Transition Diagram

State Transition Diagram (STD) memberikan gambaran secara eksplisit mengenai definisi formal dari *behaviour* sistem. Kelemahan yang paling besar dari *state transition diagram* adalah seluruh *state* yang mungkin dari sistem harus digambarkan. Untuk sistem yang kompleks, state tumbuh secara eksponensial. Hal ini tentunya akan membuat STD terlalu kompleks jika sistem yang dibuat adalah sistem berskala besar (Scotts, 2000). Tabel 2.4 dibawah ini berisi notasi-notasi yang digunakan untuk menggambarkan STD.

Tabel 2.4 Notasi di *State Transition Diagram* (STD)

No	Gambar	Nama	Keterangan
1		*State*	Merepresentasikan *state* atau kondisi sistem
2		*Transition*	Untuk merepresetasikan transisi antar *state*. Pada *transition* terdapat keterangan yang terbagi menjadi kondisi dan aksi.

Gambar 2.7 dibawah ini merupakan contoh *State Transition Diagram* dari permainan catur.



Gambar 2.6 Contoh *State Transition Diagram*

Sumber : Dalbey (2017)

2.6 Teknologi Pengembangan Sistem

2.6.1 Node.js

Node.js adalah *framework* Javascript yang sangat powerful yang dibangun diatas mesin Javascript Google Chrome v8. Node.js menggunakan *event-driven*, *non-blocking I/O model* yang membuatnya *lightweight* dan efisien. Node.js memiliki *package ecosystem* yang bernama npm. Dengan menggunakan npm, *developer* dimudahkan untuk meng*install packages* yang diperlukannya hanya dengan menggunakan Command Prompt. Gambar 2.8 dibawah ini merupakan *script* contoh pembuatan *server* Node.js.

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Gambar 2.7 Contoh *Script* Node.js

2.6.1.1 Package-package Node.js yang Digunakan

a) Passport.js

Passport merupakan suatu *middleware* authentikasi untuk Node.js. Passport menyediakan ratusan *strategy* mulai dari *local strategy*, Facebook, Twitter, Gmail dan lainnya. Gambar 2.9 dibawah ini merupakan perintah untuk mengunduh *package* Passport.js melalui npm.

```
$ npm install passport
```

Gambar 2.8 Cara *Install* Passport.js

Sumber: (Passport, 2017)

b) Express-mysql-session

Merupakan suatu *package* yang digunakan untuk membantu pengimplementasian *persistent session*. Karena *session* di Node.js disimpan di *memory*, maka data *session* akan hilang jika *server* di*restart* atau ketika *server crash*. Hal ini mengakibatkan pengguna yang sedang *login* saat itu dikeluarkan dari sistem. Dengan menggunakan *package* ini, setelah user login maka data *session* otomatis akan dimasukkan kedalam suatu tabel di *database*. Gambar

2.10 dibawah ini merupakan contoh pengimplementasian *package* express-mysql-session.

```
var session = require('express-session');
var MySQLStore = require('express-mysql-session')(session);

var options = {
    host: 'localhost',
    port: 3306,
    user: 'session_test',
    password: 'password',
    database: 'session_test',
    schema: {
        tableName: 'custom_sessions_table_name',
        columnNames: {
            session_id: 'custom_session_id',
            expires: 'custom_expires_column_name',
            data: 'custom_data_column_name'
        }
    }
};

var sessionStore = new MySQLStore(options);
```

Gambar 2.9 Konfigurasi Express-mysql-session

Sumber: (npm, 2017)

c) Node-google-calendar

Node-google-calendar adalah suatu *package* yang digunakan untuk mempermudah pengimplementasian Google Calendar API dengan menggunakan Node.js. Gambar 2.11 berikut ini merupakan contoh pengimplementasian Google Calendar API menggunakan *package* node-google-calendar.

```
const KEYFILE = '<yourpem.pem>';
const SERVICE_ACCT_ID = '<service_account>@<project_name>.iam.gservic
const CALENDAR_ID = {
  'primary': '<main-calendar-id>@gmail.com',
  'calendar-1': 'calendar1@group.calendar.google.com',
  'calendar-2': 'calendar2@group.calendar.google.com'
};
const TIMEZONE = 'UTC+08:00';

module.exports.keyFile = KEYFILE;           //or if using json keys -
module.exports.serviceAcctId = SERVICE_ACCT_ID;
module.exports.calendarId = CALENDAR_ID;
module.exports.timezone = TIMEZONE;
```

Gambar 2.10 Konfigurasi node-google-calendar

Sumber: (npm, 2017)

Gambar 2.12 dibawah ini merupakan kode yang digunakan untuk me*load* informasi-informasi *credentials* yang digunakan untuk proses authentikasi dalam menggunakan Google Calendar API.

```
const CONFIG = require('./config/Settings');
const CalendarAPI = require('node-google-calendar');
let cal = new CalendarAPI(CONFIG);
```

Gambar 2.11 Kode Import Konfigurasi node-google-calendar

Sumber: (npm, 2017)

Gambar 2.13 berikut ini merupakan contoh *script* yang digunakan untuk memasukkan event kedalam suatu kalender yang spesifik pada aplikasi Google Calendar.

```
let params = {
    'start': { 'dateTime': '2017-05-20T07:00:00+08:00' },
    'end': { 'dateTime': '2017-05-20T08:00:00+08:00' },
    'location': 'Coffeeshop',
    'summary': 'Breakfast',
    'status': 'confirmed',
    'description': '',
    'colorId': 1
};

cal.Events.insert(calendarId, params)
  .then(resp => {
    console.log('inserted event:');
    console.log(resp);
  })
  .catch(err => {
    console.log('Error: insertEvent-' + err.message);
  });
```

Gambar 2.12 Contoh *Insert* Menggunakan node-google-calendar

Sumber: (npm, 2017)

d) Nodemon

Nodemon merupakan *tool* yang dapat digunakan untuk membantu menjalankan aplikasi berbasis Node.js dan memonitor perubahan yang terjadi pada file (misal .js dan .html) kemudian saat terjadi perubahan, nodemon akan me*restart server*. Sehingga perubahan tersebut dapat langsung dieksekusi dan ditampilkan. Untuk menginstall Node.js dapat dilakukan menggunakan npm dengan mengetikkan *script* pada Gambar 2.14.

```
npm install -g nodemon
```

Gambar 2.13 Cara *Install* Nodemon

Sumber : (GitHub, 2017)

2.6.2 Express.js

ExpressJS adalah *framework* Node.js yang minimalis dan fleksibel yang menyediakan kumpulan fitur-fitur yang *robust* untuk membuat aplikasi web dan *mobile* (ExpressJS, 2017). ExpressJS menyediakan *interface* yang minimal untuk membangun aplikasi. ExpressJS sangat fleksibel karena modul-modulnya tersedia di npm dan dapat langsung diinstall menggunakan Command Prompt.

Untuk dapat menggunakan Express, diperlukan Node.js yang telah terinstall di komputer. Express dapat di*install* dengan menggunakan npm. Npm merupakan *package manager* dari Node.js. Express.js tidak memiliki struktur program yang *strict*. Programmer bebas membuat struktur program sesuai keinginannya. Express juga menyediakan fitur *express generator* yang dapat diinstall menggunakan npm. Gambar 2.15 dibawah ini merupakan ilustrasi struktur program yang di*generate* menggunakan *express generator*.

Gambar 2.14 Struktur Program Express

Sumber: (Express.js, 2017)

File www yang terletak didalam *folder* bin merupakan *entry point* dari aplikasi. File www akan dipanggil pertama kali ketika program dijalankan. Directory *view* berisi seluruh *view* dari aplikasi. Direktori *route* berisi *route* serta juga dapat berisi *controller* dan *model* dari aplikasi. File package.json merupakan file json yang berisi *dependencies* dari aplikasi. Yang terakhir adalah folder public yang berisi file CSS *external*, javascript, gambar serta *libraries* yang digunakan.

2.6.3 jQuery

jQuery adalah *library* Javascript yang cepat dan kaya akan fitur. jQuery merupakan salah satu *framework* Javascript yang digunakan untuk mempermudah pengimplementasian Javascript didalam pembuatan aplikasi. jQuery menyederhanakan banyak *task* yang cukup rumit jika diimplementasikan langsung menggunakan Javascript seperti melakukan AJAX (Asynchronous Javascript and XML) *calls* dan manipulasi DOM (*Document Object Model*)**.** Gambar 2.16 dibawah ini merupakan *script* yang berfungsi untuk menyembunyikan konten HTML yang memiliki atribut *class* yang bernama “target”.

```
1 | $( ".target" ).hide();
```

Gambar 2.15 Contoh Kode jQuery

2.6.4 SemanticUI

SemanticUI merupakan *framework* CSS yang digunakan untuk mempercepat proses pembangunan *User Interface* dari suatu aplikasi. SemanticUI menyediakan berbagai macam elemen mulai dari elemen yang sederhana seperti *button* hingga elemen kompleks seperti *accordion, modal dan sidebar* yang tentunya akan memakan waktu jika dibangun dari awal. SemanticUI menggunakan frasa sederhana yang mudah dipahami untuk men*trigger* setiap fungsionalitasnya. Gambar 2.17 dibawah ini merupakan proses inisialisasi untuk men*trigger* modal pada SemanticUI menggunakan bantuan *framework* lain yaitu jQuery.

```
$('.ui.modal')
  .modal('show')
;
```

Gambar 2.16 Contoh Kode jQuery SemanticUI

Sumber: (SemanticUI, 2017)

Gambar 2.18 dibawah ini merupakan kode HTML yang digunakan untuk membuat *modal* di SemanticUI.

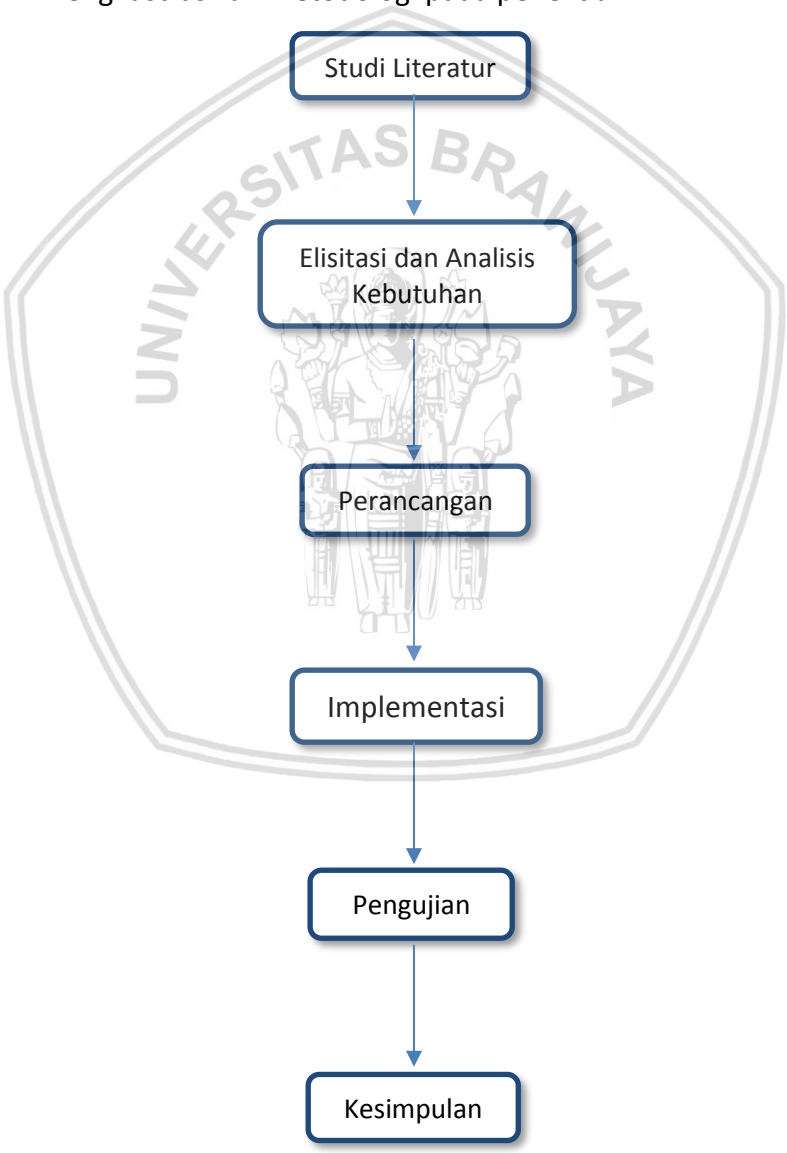
```
<div class="ui modal">
  <i class="close icon"></i>
  <div class="header">
    Profile Picture
  </div>
  <div class="image content">
    <div class="ui medium image">
      <img src="/images/avatar/large/chris.jpg">
    </div>
    <div class="description">
      <div class="ui header">We've auto-chosen a profile image for you.</div>
      <p>We've grabbed the following image from the <a
href="https://www.gravatar.com" target="_blank">gravatar</a> image associated with
your registered e-mail address.</p>
      <p>Is it okay to use this photo?</p>
    </div>
  </div>
  <div class="actions">
    <div class="ui black deny button">
      Nope
    </div>
    <div class="ui positive right labeled icon button">
      Yep, that's me
      <i class="checkmark icon"></i>
    </div>
  </div>
</div>
```

Gambar 2.17 Contoh Kode HTML SemanticUI

Sumber: (SemanticUI, 2017)

BAB 3 METODOLOGI PENELITIAN

Pada bab Metodologi Penelitian, penulis akan membahas langkah-langkah yang akan dilakukan penulis untuk menyelesaikan permasalahan yang diangkat pada penelitian ini. Pada penelitian ini, penulis akan menggunakan SDLC (*Software Development Life Cycle*) *Waterfall*. Metodologi penelitian pada penelitian ini dimulai dari studi literatur. Kemudian, karena penulis menggunakan pendekatan *Object Oriented*, pada tahap elisitasi dan analisis kebutuhan penulis akan menggunakan metode OOA *(Object Oriented Analysis)*, tahap perancangan menggunakan metode OOD *(Object Oriented Design)* dan tahap implementasi menggunakan metode OOP *(Object Oriented Programming)* dan dilanjutkan dengan tahap pengujian dan penarikan kesimpulan. Gambar 3.1 dibawah ini mengilustrasikan metodologi pada penelitian ini.



Gambar 3.1 Metodologi Penelitian

3.1 Studi Literatur

Pada tahap studi literatur penulis akan membahas kajian teoritis dan empiris yang digunakan sebagai dasar teori. Pada tahap ini, penulis akan melakukan pencarian referensi serta pemahaman teori dan teknologi yang relevan didalam penelitian ini. Referensi teori dan teknologi yang digunakan didapat dari buku, artikel *online*, jurnal dan *website*. Dalam penelitian ini, akan dilakukan pencarian referensi yang berisi:

a. Pengetahuan lebih lanjut tentang OOA, OOD, OOP.
b. Pengetahuan lebih lanjut tentang Javascript.
c. Pengetahuan lebih lanjut tentang *Object Oriented* Javascript (OOJS) pada ES6.
d. Pemahaman konsep *asynchronous programming*.
e. Pemahaman konsep *callback*, *promise* dan *async await* di Javascript.
f. Pemahaman lebih lanjut terkait *framework* CSS yang akan digunakan yaitu SemanticUI.
g. Pengetahuan lebih lanjut terkait teknologi yang akan digunakan didalam penelitian ini yaitu *platform* Node.js dan *framework* Express.js.
h. Dokumentasi Google Calendar API untuk *platform* Node.js.
i. *Package-package* Node.js yang akan digunakan untuk menunjang pembuatan aplikasi.
j. *Security Best Practice* dalam *framework* Express.js.

3.2 Elisitasi dan Analisis Kebutuhan

Elisitasi dan analisis kebutuhan merupakan langkah pertama yang dilakukan dalam SDLC *Waterfall.* Karena menggunakan SDLC *Waterfall* maka pada tahap ini harus menghasilkan definisi dan spesifikasi kebutuhan yang lengkap, *clear* dan *fixed.* Langkah-langkah detail yang akan dilakukan penulis pada tahap elisitasi dan analisis kebutuhan yaitu:

a. Elisitasi atau Penggalian Kebutuhan

   Untuk menggali kebutuhan, penulis akan menggunakan metode *interview* secara langsung kepada Mahasiswa, Wakil Dekan 2, Wakil Dekan 3, Kepala/*Staff* TU Subbag Umum dan Perlengkapan, Kepala/*Staff* TU Subbag Keuangan dan Kepala Lab Sistem Informasi Manajemen Fakultas Ilmu Administrasi Universitas Brawijaya. Pertanyaan-pertanyaan yang akan diajukan dalam *interview* antara lain proses bisnis atau prosedur pengajuan *event*, pencairan dana dan peminjaman *venue* yang berlaku di Fakultas Ilmu Administrasi UB, masalah-masalah yang timbul dalam pengajuan *event* di FIA serta data lain seperti daftar *venue* yang biasa digunakan untuk penyelenggaraan *event* di FIA UB.

b. Analisis Kebutuhan

Kemudian dari hasil *interview* pada tahap elisitasi kebutuhan, penulis akan melakukan proses analisis yang akan menghasilkan calon-calon aktor sistem, definisi dan spesifikasi kebutuhan sistem secara lengkap, baik kebutuhan fungsional maupun kebutuhan non-fungsional. Kebutuhan fungsional digunakan untuk menetapkan fitur-fitur apa saja yang harus ada di aplikasi (apa yang dapat dilakukan aplikasi). Sedangkan kebutuhan non-fungsional digunakan untuk menetapkan batasan layanan dari aplikasi.

c. Verifikasi dan Validasi Kebutuhan

Verifikasi kebutuhan bertujuan untuk memastikan bahwa prosedur pengajuan *event* telah dibuat dengan benar (telah memenuhi parameter verifikasi). Sedangkan validasi kebutuhan bertujuan untuk memastikan penulis telah membuat prosedur pengajuan *event* yang benar (telah memenuhi parameter validasi). Pada tahap ini, penulis akan membuat dokumen terpisah yang berisi *flowchart* yang mengilustrasikan prosedur pengajuan *event* organisasi mahasiswa dan prosedur pengajuan *event* unit internal fakultas yang didapat dari hasil elisitasi dan analisis kebutuhan. Kemudian penulis akan melakukan validasi prosedur pengajuan *event* tersebut ke Kepala Gugus Jaminan Mutu (GJM) Fakultas Ilmu Administrasi Universitas Brawijaya. Penulis juga melakukan verifikasi kebutuhan untuk memastikan bahwa kebutuhan telah dengan benar didefinisikan dan dispesifikasikan dengan cara memeriksa kebutuhan menggunakan parameter-parameter verifikasi itu sendiri.

d. Manajemen Kebutuhan

Tahap ini dilakukan dengan cara memberi kode untuk setiap definisi dan spesifikasi kebutuhan sehingga memudahkan identifikasi setiap definisi dan spesifikasi kebutuhan.

Setelah itu, dari definisi dan spesifikasi kebutuhan yang telah diverifikasi dan divalidasi tersebut akan menjadi dasar bagi penulis untuk membuat pemodelan kebutuhan sistem. Karena penulis menggunakan metode OOA (*Object Oriented Analysis*), untuk memodelkan kebutuhan sistem penulis akan membuat *Use Case Diagram* dan *Use Case Scenario* serta *State Transition Diagram*.

State Transition Diagram (STD) digunakan untuk memodelkan alur dan status disposisi proposal atau persetujuan *event*. Pada tahap analisis kebutuhan ini penulis juga akan membuat *Business Process Modelling* (BPM) diagram untuk memodelkan prosedur pengajuan *event* yang sedang berjalan dan prosedur yang akan digunakan pada sistem yang akan dibuat. Tabel 3.1 berikut ini merupakan aplikasi-aplikasi yang akan digunakan untuk menunjang tahap analisis kebutuhan pada penelitian ini.

Tabel 3.1 Aplikasi Penunjang Analisis Kebutuhan

No	Diagram/Komponen yang Dibuat	Aplikasi yang Digunakan
1	*Business Process Modelling* (BPM)	draw.io
2	*Use Case Diagram*	Visual Paradigm *Community Edition*

3.3 Perancangan

Perancangan merupakan tahap kedua didalam SDLC *Waterfall.* Perancangan berfungsi untuk memodelkan suatu perancangan sistem menggunakan diagram-diagram yang sesuai dengan paradigma pemrograman yang digunakan. Karena didalam penelitian ini penulis menggunakan paradigma pemrograman berorientasi objek, sehingga dalam memodelkan perancangan sistem penulis akan menggunakan Diagram UML (*Unified Modelling Language*). Perancangan sistem akan dibagi menjadi 4 bagian yaitu:

a. Perancangan Arsitektur

   Perancangan arsitektur berisi *Class Diagram* yang menggambarkan kelas, struktur kelas (atribut dan *method*) dan relasi antar kelas serta *Sequence Diagram* yaitu diagram yang menggambarkan interaksi antar objek didalam sistem.

b. Perancangan Komponen

   Pada perancangan komponen, penulis akan membuat *pseudocode* dari beberapa *method* yang ada pada aplikasi Manajemen *Event*.

c. Perancangan Data

   Perancangan data berisi *Entity Relationship Diagram* (ERD). *Entity Relationship Diagram* dibuat murni berdasarkan definisi dan spesifikasi kebutuhan sistem dan belum memikirkan detail dari implementasi data. ERD akan digunakan sebagai dasar untuk membuat *Physical Data Model* (PDM) pada tahap implementasi.

d. Perancangan *User Interface* (UI)

   Pada tahap ini penulis akan membuat perancangan UI (*User Interface*) dari beberapa halaman yang ada di aplikasi Manajemen *Event*.

Tabel 3.2 berikut ini merupakan aplikasi-aplikasi yang akan digunakan untuk menunjang tahap peracangan pada penelitian ini.

Tabel 3.2 Aplikasi Penunjang Perancangan

No	Diagram/Komponen yang Dibuat	Aplikasi yang Digunakan
1	*Sequence Diagram, Class Diagram, ERD*	Visual Paradigm *Community Edition*
2	Perancangan *User Interface*	Moqups (https://app.moqups.com)

3.4 Implementasi

Tahap implementasi merupakan tahap keempat didalam SDLC *Waterfall.* Implementasi bertujuan untuk menginterpretasikan hasil perancangan kedalam kode program sehingga akan menghasilkan suatu produk perangkat lunak. Pada penelitian ini, penulis akan menggunakan Node.js dengan bantuan *framework* Express untuk mengimplementasikan *back-end* aplikasi ini. Sedangkan untuk *front-end*, penulis akan menggunakan *framework* CSS yaitu SemanticUI. Karena pada penelitian ini penulis menggunakan aplikasi Google Calendar untuk menyimpan data peminjaman *venue*, untuk mengimplementasikan Google Calendar API penulis akan menggunakan *package* node-google-calendar. Pada tahap implementasi, penulis akan mendefinisikan spesifikasi *hardware* dan *software* yang digunakan didalam menunjang penelitian ini. Selain itu pada tahap ini penulis akan menginterpretasikan perancangan komponen (*pseudocode*) menjadi kode *back-end* dan perancangan UI menjadi kode *front-end*.

Pengkodean program secara garis besar akan dibagi menjadi 2 bagian yaitu pengkodean *front-end* dan pengkodean *back-end*. Detail dari proses implementasi adalah sebagai berikut:

a. Pengkodean *front-end* menggunakan HTML dengan bantuan *framework* CSS yaitu Semantic UI. Hasil dari pengkodean *front-end* akan diletakkan di folder *views*.
b. Pembuatan skema *database* dan penggunaan *server database* MySQL dilakukan menggunakan bantuan *software* XAMPP.
c. Pengkodean *back-end* dengan menggunakan Express.js yaitu merupakan *framework* NodeJS. Hasil dari pengkodean *back-end* akan diletakkan di folder *root*, *routes*, *models*, *helpers* dan *controllers*.
d. Konfigurasi dari masing-masing *package* yang digunakan di aplikasi ini akan diletakkan di folder *configs*.

Tabel 3.3 berikut ini merupakan aplikasi-aplikasi yang akan digunakan untuk menunjang tahap implementasi pada penelitian ini.

Tabel 3.3 Aplikasi Penunjang Implementasi

No	Diagram/Komponen yang Dibuat/Digunakan	Aplikasi yang Digunakan
1	ERD (*Physical Data Model*)	XAMPP
2	Implementasi UI (*front-end*)	Visual Studio Code
3	Kode Program (*back-end*)	Visual Studio Code
4	Skema Database	XAMPP
5	RDBMS	MySQL

3.5 Pengujian

Pengujian merupakan tahapan yang bertujuan untuk memeriksa apakah seluruh kebutuhan fungsional dan non-fungsional yang telah didefinisi dan dispesifikasikan sudah berjalan sesuai harapan. Penulis akan menggunakan metode *manual testing* (tidak menggunakan *tools*) untuk menguji kebutuhan fungsional pada penelitian ini. Sedangkan untuk kebutuhan non-fungsional akan diuji secara *automate* (menggunakan *tools*). Fase pengujian pada penelitian ini akan dibagi menjadi 3 tahap yaitu:

a. Pengujian Unit
   Pengujian unit dilakukan untuk menguji kebutuhan fungsional dari aplikasi Manajemen *Event*. Pada pengujian unit, penulis akan menggunakan *white box testing*. Metode yang akan digunakan yaitu *basis path testing*.

b. Pengujian Sistem/Validasi

   Pengujian validasi dilakukan untuk menguji kebutuhan fungsional dari aplikasi Manajemen *Event*. Pada pengujian validasi atau pengujian sistem, penulis akan menggunakan metode *scenario-based testing.* Penulis akan menguji setiap definisi kebutuhan yang ada serta skenario alternatif dari masing-masing definisi kebutuhan yang terdapat di *use case scenario*.

c. Pengujian *Browser Compatibility*
   Pengujian browser compatibility dilakukan untuk menguji kebutuhan non-fungsional dari aplikasi Manajemen *Event.* Pada *Browser Compatibility Testing*, penulis akan menggunakan metode *automate testing* dengan *tools* bernama SortSite.

Tabel 3.4 Aplikasi Penunjang Pengujian

No	Diagram/Komponen yang Dibuat/Digunakan	Aplikasi yang Digunakan
1	*Graph* pada *basic path testing*	draw.io
2	*Browser Compatibility Testing*	SortSite

3.6 Kesimpulan

Proses penarikan kesimpulan ini akan dilakukan setelah semua fase dimulai dari elisitasi dan analisis kebutuhan sampai pengujian selesai dilakukan. Kesimpulan diperoleh dari hasil pengujian perangkat lunak. Hal ini digunakan untuk menentukan apakah perangkat lunak sudah layak atau belum untuk di antarkan dan digunakan oleh pengguna. Pada tahap ini juga terdapat saran yang menjelaskan tentang peluang pengembangan selanjutnya dari penelitian ini.

BAB 4 ANALISIS KEBUTUHAN

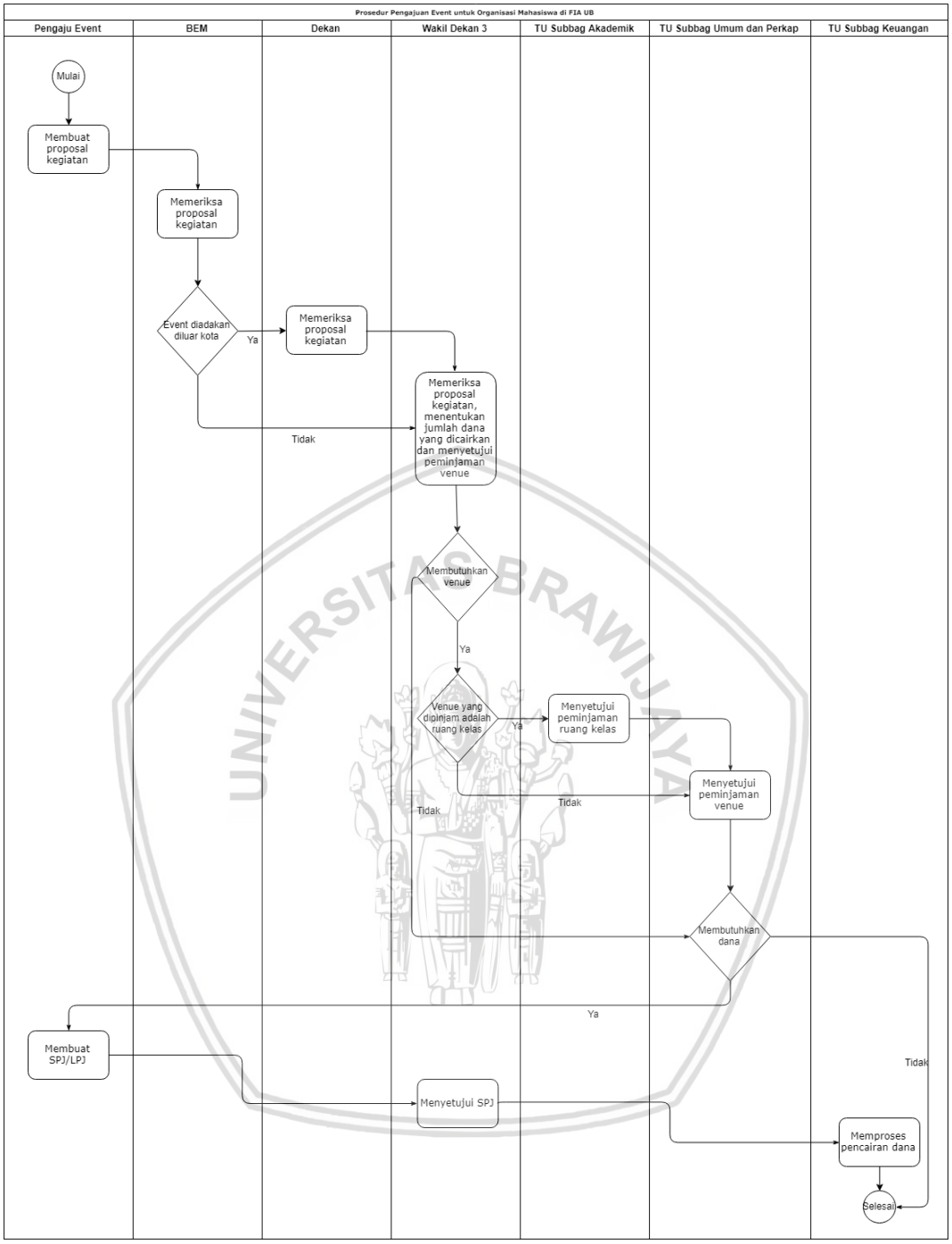
Pada bab ini akan membahas proses elisitasi dan analisis kebutuhan. Penulis menggunakan teknik wawancara atau *interview* secara langsung kepada beberapa *stakeholder* pengajuan *event*, pencairan dana dan peminjaman *venue* di Fakultas Administrasi Universitas Brawijaya. Setelah data telah terkumpul kemudian pada bab ini akan dilakukan analisis. Karena pada penelitian ini penulis menggunakan pendekatan *object oriented*, maka pada tahap analisis kebutuhan ini penulis akan menggunakan metode OOA (*Object Oriented Analysis*). Pada OOA penulis akan melakukan identifikasi aktor sistem, identifikasi *service* apa saja yang disediakan oleh sistem (definisi kebutuhan fungsional dan non-fungsional serta spesifikasi kebutuhan fungsional sistem). Selain itu, penulis juga akan mengidentifikasi kandidat objek dan kelas yang akan digunakan pada tahap selanjutnya yaitu OOD (*Object Oriented Design*). Setelah itu akan dilakukan proses verifikasi dan validasi kebutuhan. Setelah kebutuhan diverifikasi dan divalidasi akan dilanjutkan dengan proses pemodelan kebutuhan dengan menggunakan Diagram *Use Case* dan *Use Case Scenario*. Kemudian juga akan dibuat *State Transition Diagram* untuk memodelkan alur dan status disposisi proposal atau persetujuan event.

4.1 Analisis dan Elisitasi Kebutuhan

Pada tahap elisitasi, penulis mengelisitasi kebutuhan dengan menggunakan teknik *interview* atau wawancara secara langsung. Penulis telah melakukan wawancara kepada Wakil Dekan 2 (Dr. Hamidah Nayati Utami, M.Si), Wakil Dekan 3 (Dr. Mohammad Rozikin, M.AP), Staff TU Umum dan Perlengkapan (Dyah Susanti), Kepala TU Subbag Keuangan (Jaedi, SP), Kepala Lab Sistem Informasi Manajemen (Rizky Yudhi Dewantara, S.Sos., M.AP) dan Mahasiswa Fakultas Ilmu Administrasi Universitas Brawijaya (Aria Adi Negoro dan Aan Suryana).

Penulis telah melakukan wawancara kepada Wakil Dekan 2 untuk mengetahui alur pengajuan *event* jika pengaju *event* adalah unit internal fakultas. Sedangkan untuk mengetahui alur pengajuan *event* organisasi mahasiswa, penulis melakukan wawancara kepada Wakil Dekan 3. Untuk mengetahui prosedur peminjaman fasilitas fakultas khususnya peminjaman *venue*, penulis telah melakukan wawancara terhadap *staff* Tata Usaha Subbag Umum dan Perlengkapan.

Kemudian penulis melakukan wawancara kepada Kepala Tata Usaha Subbag Keuangan Keuangan untuk mengetahui prosedur pencairan dana. Selain itu, penulis juga mewawancarai Kepala Lab SIM FIA untuk mengetahui permasalahan yang timbul dari pengajuan event secara manual dari sisi pengaju event (unit internal fakultas). Kemudian penulis juga mewawancarai mahasiswa FIA UB untuk mengetahui masalah yang timbul dari pengajuan *event* secara manual dari perspektif pengaju *event* dalam hal ini organisasi mahasiswa. Hasil wawancara secara detail digambarkan menggunakan *Business Process Modelling* dibawah ini.



Gambar 4.1 BPM Prosedur Pengajuan *Event* untuk Organisasi Mahasiswa

Berikut ini merupakan penjelasan BPM prosedur pengajuan *event* untuk organisasi mahasiswa (Gambar 4.1):

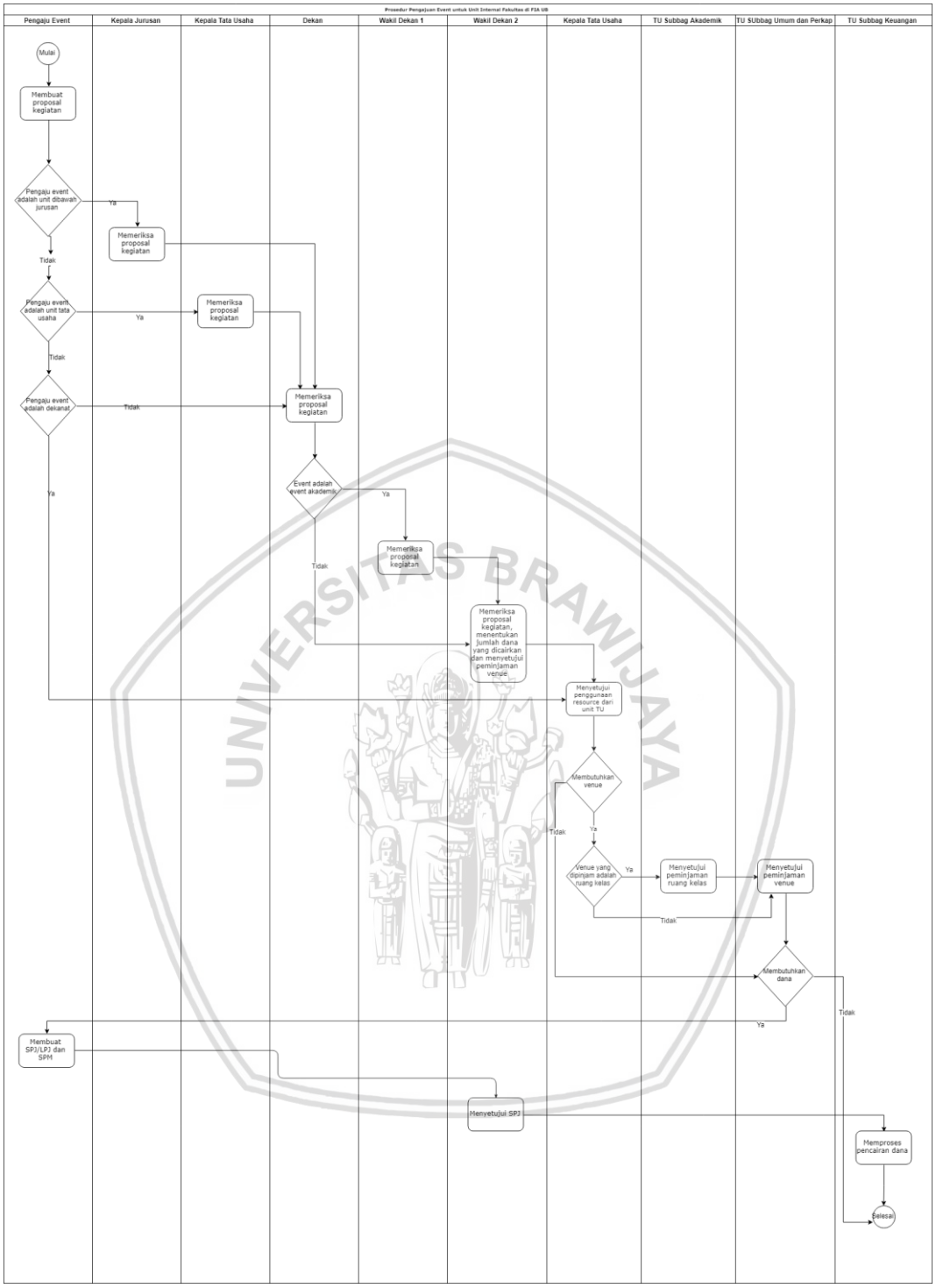
1. Pengaju *event* dalam hal ini organisasi mahasiswa mengajukan *event* ke Badan Eksekutif Mahasiswa (BEM) dengan menyertakan proposal *event*.
2. Selanjutnya, jika *event* diadakan di luar kota Malang, proposal harus mendapatkan persetujuan dari Dekan.

3. Setelah itu, *event* harus disetujui oleh Wakil Dekan 3. Jika event membutuhkan dana maka Wakil Dekan 3 akan menentukan berapa jumlah dana yang akan dicairkan. Wakil Dekan 3 juga mengizinkan pengajuan peminjaman *venue* untuk penyelenggaraan *event* ke TU Subbag Umum dan Perlengkapan.
4. Kemudian pengaju *event* dapat meminjam *venue* di bagian TU Subbag Umum Perlengkapan. Pengaju *event* diharuskan mengisi *form* peminjaman *venue* yang dapat dilihat pada Lampiran E dan Lampiran F. Prosedur peminjaman venue secara detail dapat dilihat pada *flowchart* yang dapat terdapat pada Lampiran D.
5. Jika *event* membutuhkan dana maka pengaju event diharuskan membuat SPJ/LPJ untuk mencairkan dana ke TU Subbag Keuangan. Prosedur pencairan dana secara detail dapat dilihat pada Lampiran B dan Lampiran C.

Gambar 4.1 diatas merupakan *Business Process Modelling* (BPM) yang dibuat berdasarkan hasil wawancara penulis dengan Wakil Dekan 3. Dari penuturan Mahasiswa FIA UB, mahasiswa sering kesusahan untuk mengetahui status disposisi proposal *event* yang diajukannya. Selain itu, proses persetujuan proposal *event* menjadi terhambat apabila pihak-pihak yang berperan menyetujui pengajuan *event* dalam hal ini Dekanat, Kepala Jurusan atau KTU tidak ada di tempat dikarenakan sedang bertugas keluar kota atau karena alasan lain. Penulis juga sempat bertanya kepada Wakil Dekan 2 terkait permasalahan pengajuan *event* di Fakultas Ilmu Administrasi UB. Menurut penuturan Wakil Dekan 2, banyak mahasiswa yang masih belum mengetahui alur pengajuan *event*.

Gambar 4.2 dibawah merupakan gambaran dari pengajuan *event* oleh unit internal fakultas yang dibuat berdasarkan wawancara dengan Wakil Dekan 2. Menurut Kepala Lab SIM FIA UB, unit internal fakultas di FIA sering berebut *venue* untuk penyelenggaraan *event* masing-masing unit. Oleh karena itu Kepala Lab SIM FIA UB mengharapkan agar ada sistem yang dapat digunakan untuk mereservasi *venue* untuk penyelenggaraan *event*. Berikut ini merupakan penjelasan BPM prosedur pengajuan *event* untuk unit internal fakultas (Gambar 4.2):

1. Jika pengaju *event* adalah Dekanat (Dekan, Wakil Dekan 1, Wakil Dekan 2 dan Wakil Dekan 3), maka proposal akan langsung didisposisi ke KTU.
2. Jika pengaju *event* merupakan unit dibawah jurusan misalnya program studi, maka proposal harus mendapatkan persetujuan Kepala Jurusan terlebih dahulu.
3. Jika pengaju *event* merupakan unit Tata Usaha, maka proposal harus mendapat persetujuan KTU terlebih dahulu.
4. Selanjutnya proposal akan didisposisi ke Dekan.



Gambar 4.2 BPM Prosedur Pengajuan *Event* Untuk Unit Internal Fakultas

5. Setelah Dekan menyetujui, jika *event* merupakan *event* akademik, maka pengajuan *event* membutuhkan persetujuan Wakil Dekan 1.
6. Setelah Wakil Dekan 1 menyetujui, selanjutnya proposal akan didisposisi ke Wakil Dekan 2.

7. Setelah itu, *event* harus disetujui oleh Wakil Dekan 2. Jika *event* membutuhkan dana maka Wakil Dekan 2 akan menentukan berapa jumlah dana yang akan dicairkan. Wakil Dekan 2 juga mengizinkan pengajuan peminjaman *venue* untuk penyelenggaraan *event* ke TU Subbag Umum dan Perlengkapan.
8. Setelah itu proposal akan didisposisi ke KTU. Setelah disetujui KTU, kemudian pengaju *event* dapat meminjam *venue* di bagian TU Subbag Umum Perlengkapan dan mencairkan dana di bagian TU Subbag Keuangan. Pengaju *event* diharuskan mengisi *form* peminjaman *venue* yang dapat dilihat pada Lampiran E dan Lampiran F. Prosedur peminjaman venue secara detail dapat dilihat pada *flowchart* yang dapat terdapat pada Lampiran D.
9. Prosedur pencairan dana secara detail dapat dilihat pada Lampiran B dan Lampiran C. Unit internal fakultas selain mengurus SPJ/LPJ juga diharuskan mengurus SPM (Surat Perintah Membayar). Contoh SPM dapat dilihat pada Lampiran A dan prosedur pemrosesan SPM dapat dilihat pada Lampiran B.

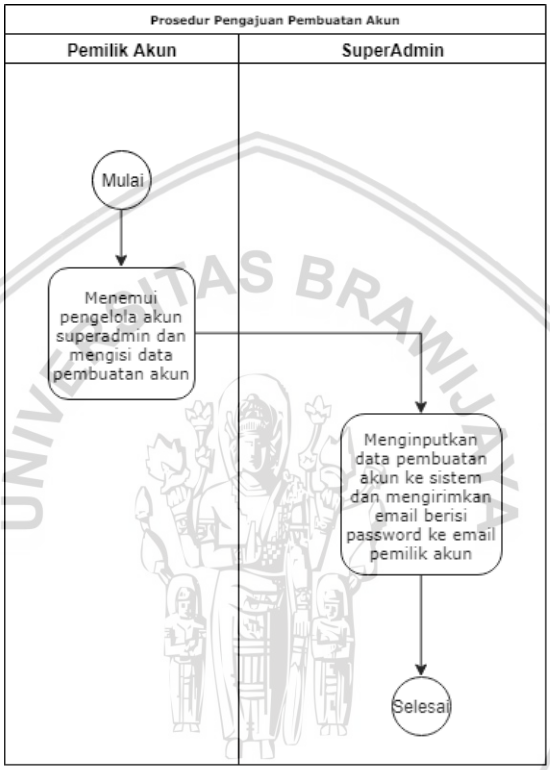
Kemudian penulis melakukan wawancara kepada *Staff* TU Subbag Umum dan Perlengkapan. Menurut *Staff* TU Subbag Umum dan Perlengkapan, sering terjadi bentrok peminjaman *venue event*. Sehingga *staff* TU Subbag Umum dan Perlengkapan mengharapkan akan ada sistem yang dapat menampilkan status peminjaman *venue* dan mengecek ketersediaan *venue*. Selain itu, *staff* TU Umum dan Perlengkapan mengatakan bahwa unit internal fakultas dapat menggeser peminjaman *venue* dari organisasi mahasiswa jika suatu unit internal fakultas ingin menyelenggarakan *event* di *venue* yang sebelumnya sudah direservasi oleh organisasi mahasiswa.

Jika demikian, maka *Staff* TU Subbag Umum dan Perlengkapan biasanya menginformasikan hal tersebut kepada organisasi mahasiswa secara langsung atau melalui telepon. Sehingga aplikasi ini juga diharapkan akan membantu menotifikasi peminjam *venue* secara otomatis apabila peminjaman *venue*nya diubah sewaktu-waktu. Selain itu menurut pendapat pribadi penulis, pencatatan peminjaman venue oleh TU Subbag Umum dan Perlengkapan (dapat dilihat pada Lampiran G) kurang efektif khususnya untuk mencatat peminjaman *venue event* rutin. Pencatatan tidak efektif karena TU Subbag Umum dan Perlengkapan harus memasukkan data peminjaman *venue* yang sama secara berulang-ulang.

Kemudian pada Lampiran C dapat dilihat SPJ/LPJ dipisahkan dari eksemplar dan harus dikirim ke rektorat. Sehingga agar menghindari kerja dua kali (melakukan *approve* di sistem dan mentandatangani *hardcopy* SPJ/LPJ), oleh karena itu persetujuan SPJ/LPJ belum akan dimasukkan kedalam sistem. Kebutuhan dari aplikasi ini diambil dari prosedur pengajuan *event* pencairan dana dan peminjaman *venue* yang berlaku di Fakultas Ilmu Administrasi Universitas Brawijaya yang terdapat pada gambar 4.1 – 4.2. Hasil dari proses analisis kebutuhan akan dijelaskan pada subbab-subbab selanjutnya.

4.2 Gambaran Umum Prosedur Aplikasi Manajemen *Event*

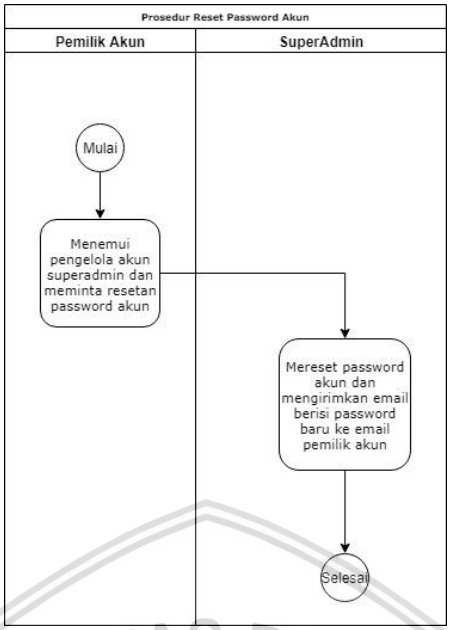
Dibawah ini merupakan diagram *Business Process Modelling Diagram* yang menggambarkan prosedur mendapatkan akun, me*reset password* akun, peminjaman *venue* dari luar FIA, pengajuan *event*, pencairan dana dan peminjaman *venue*. Prosedur pengajuan event, pencairan dana dan peminjaman venue pada aplikasi ini ini didapatkan dari *manual procedure* pengajuan *event*, pencairan dana dan peminjaman *venue* yang berlaku di Fakultas Ilmu Administrasi UB yang telah dijelaskan pada subbab sebelumnya.



Gambar 4.3 BPM Prosedur Pembuatan Akun Baru

Gambar 4.3 diatas merupakan prosedur untuk membuat akun baru pada aplikasi Manajemen *Event*. Berikut ini penjelasan secara detail terkait BPM prosedur pembuatan akun baru (Gambar 4.3):

1. Perwakilan unit/organisasi menemui pengelola akun superadmin.
2. Pembuat akun mengisi data yang diperlukan untuk pembuatan akun baru.
3. Pambuat akun mengecek email yang diisikan saat proses pembuatan akun.
4. Pembuat akun dapat *login* dengan *password* yang dikirimkan *via* email.



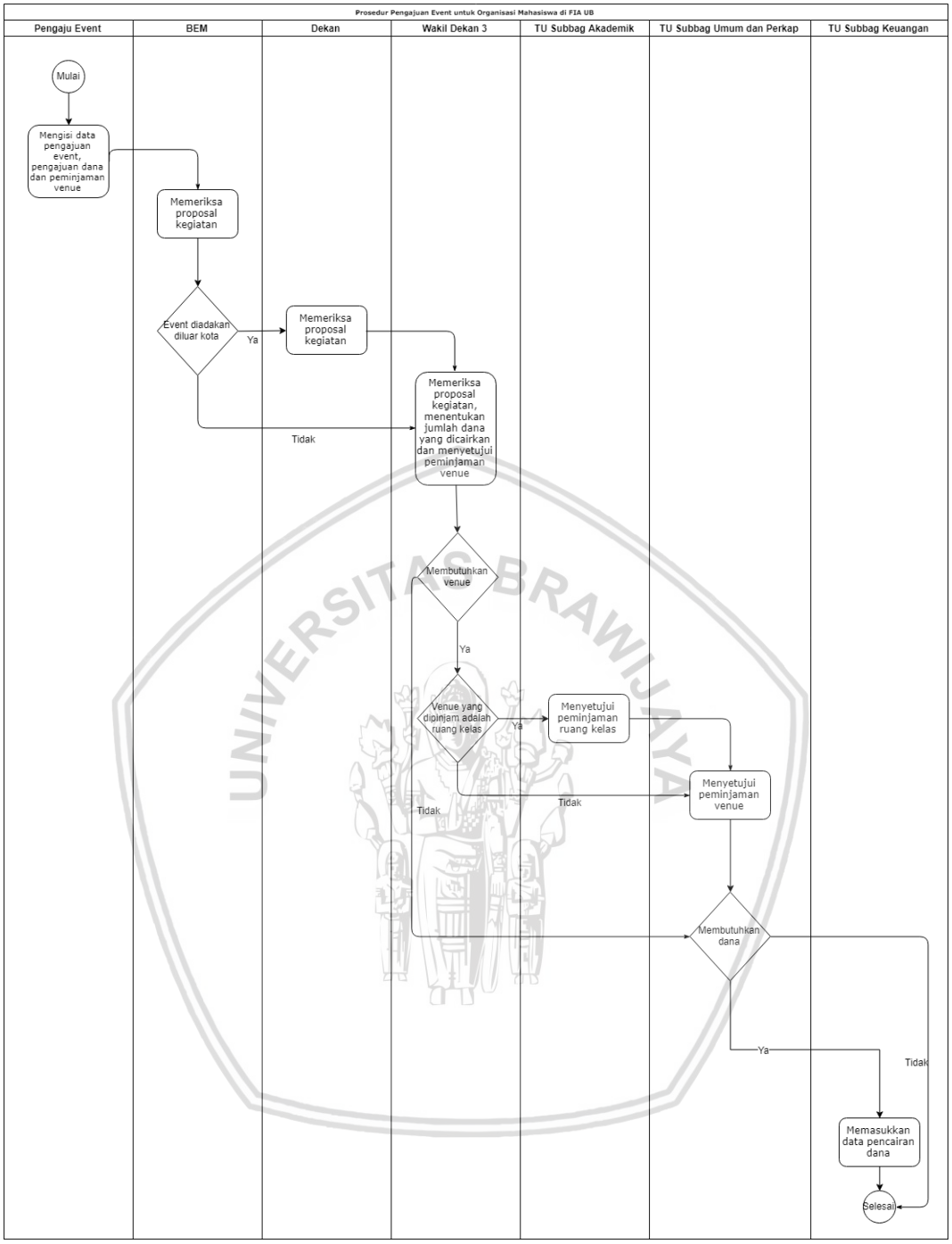
Gambar 4.4 BPM Me*reset Password* Akun

Gambar 4.4 diatas merupakan prosedur untuk me*reset password* akun pada aplikasi Manajemen *Event*. Berikut ini merupakan penjelasan dari BPM prosedur me*reset password* akun (Gambar 4.4):

1. Perwakilan unit/organisasi menemui pengelola akun superadmin.
2. Perwakilan unit/organisasi meminta pengelola akun untuk mereset akun.
3. Superadmin mengklik tombol *reset*.
4. Sistem mengirimkan email yang berisi *password* baru.
5. Pambuat akun mengecek email.
6. Pembuat akun dapat *login* dengan *password* yang dikirimkan *via* email.

Gambar 4.5 dibawah merupakan prosedur pengajuan event, peminjaman venue dan pencairan dana untuk organisasi mahasiswa. Berikut ini merupakan penjelasan BPM prosedur pengajuan *event* organisasi mahasiswa (Gambar 4.5):

1. Pertama, organisasi mahasiswa mengajukan *event* dengan mengisi data-data pengajuan *event* dan melampirkan file proposal.
2. Jika *event* membutuhkan *venue* maka sistem akan mengecek ketersediaan *venue*.
3. Jika *venue* tersedia maka *event* akan diajukan.
4. Jika *venue* tidak tersedia, maka aktor harus memilih *venue* yang lain.
5. Pertama, proposal *event* membutuhkan persetujuan BEM.
6. Jika BEM menolak maka pengaju event harus merevisi proposal event.
7. Jika *event* diadakan diluar kota, maka pengajuan *event* membutuhkan persetujuan Dekan.



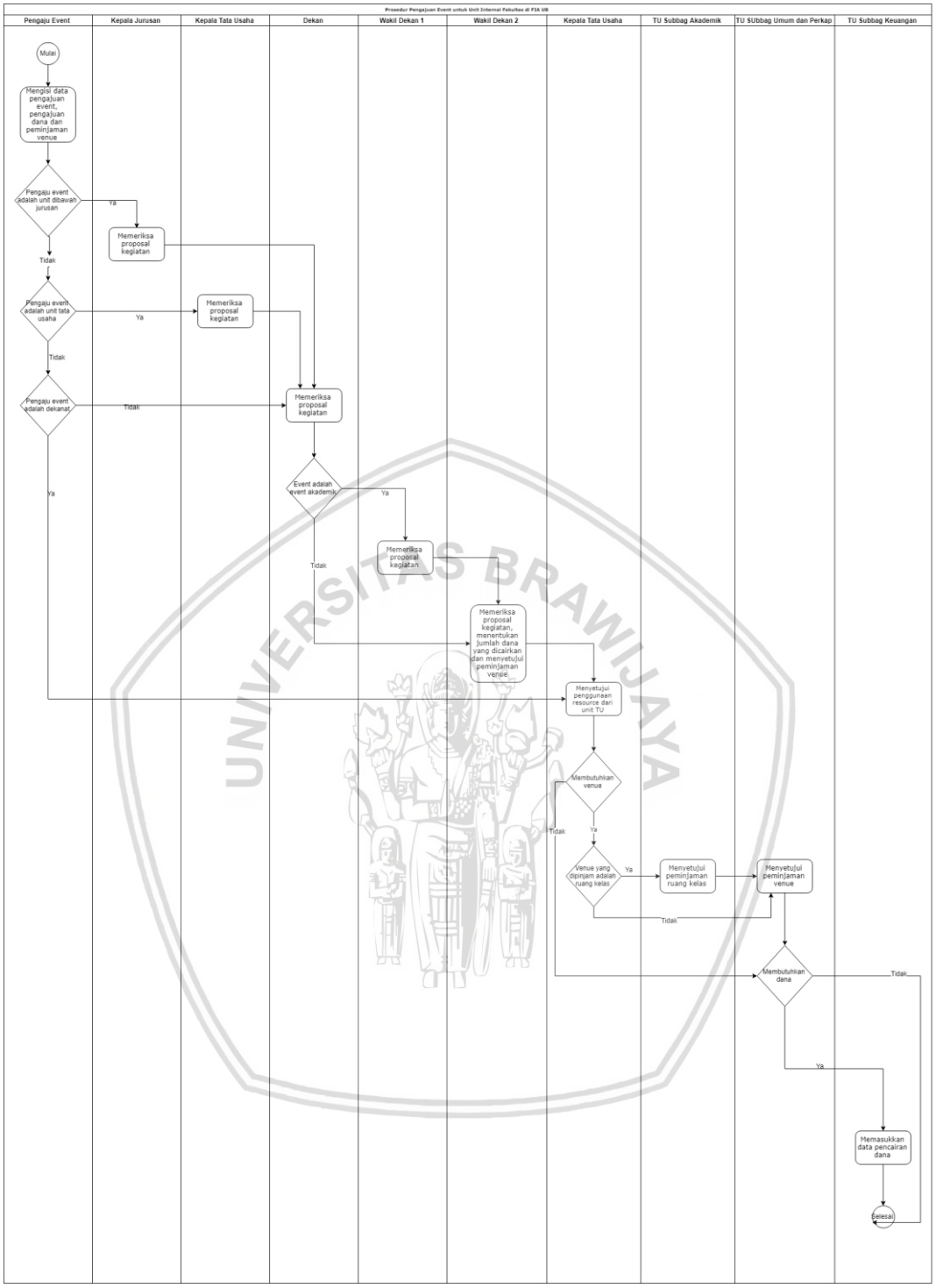
Gambar 4.5 BPM Prosedur Pengajuan *Event* Organisasi Mahasiswa

8. Jika Dekan menyetujui, selanjutnya pengajuan *event* membutuhkan persetujuan Wakil Dekan 3. Jika dana berasal dari fakultas/ kemahasiswaan maka Wakil Dekan 3 juga akan menentukan berapa jumlah dana yang akan dicairkan. Wakil Dekan 3 juga mengizinkan pengajuan peminjaman *venue*.
9. Jika dana yang disetujui kurang dari dana yang diajukan, maka sistem akan meminta konfirmasi pengaju *event* dalam hal ini organisasi mahasiswa.

10. Jika menolak, maka perwakilan organisasi mahasiswa dapat menemui Wakil Dekan 3 secara manual untuk berdiskusi lebih lanjut.
11. Jika event membutuhkan venue maka peminjaman venue memerlukan persetujuan TU Subbag Umum & Perlengkapan.
12. Kemudian, jika venue event merupakan ruang kelas, maka sebelumnya peminjaman *venue* juga memerlukan persetujuan dari TU Subbag Akademik.
13. Jika *event* membutuhkan dana, setelah LPJ sudah diserahkan dan diverifikasi serta dana telah dicairkan, maka TU Subbag Keuangan akan menginputkan data pencairan dana di sistem.

Gambar 4.6 dibawah merupakan prosedur pengajuan event, peminjaman venue dan pencairan dana untuk unit internal fakultas. Berikut ini merupakan penjelasan BPM prosedur pengajuan *event dekanat* dan unit internal fakultas (Gambar 4.6):

1. Pertama, dekanat/unit internal fakultas mengajukan *event* dengan mengisi data-data pengajuan *event* dan melampirkan file proposal.
2. Jika *event* membutuhkan *venue* maka sistem akan mengecek ketersediaan *venue*. Jika *venue* tersedia maka *event* akan diajukan. Jika *venue* tidak tersedia, maka aktor harus memilih *venue* yang lain.
3. Selanjutnya jika pengaju *event* adalah Dekanat, maka data pengajuan *event* dan proposal *event* akan langsung didisposisi ke Kepala Tata Usaha.
4. Jika pengaju *event* bukan Dekanat maka *event* membutuhkan persetujuan Dekan.
5. Sebelum itu, jika unit tersebut berada dibawah jurusan, maka harus meminta persetujuan terlebih dahulu ke Kajur.
6. Jika pengaju *event* adalah unit tata usaha maka harus meminta persetujuan ke KTU.
7. Jika *event* merupakan *event* akademik, maka pengajuan *event* membutuhkan persetujuan Wakil Dekan 1.
8. Jika *event* merupakan event non akademik, maka data pengajuan *event* dan proposal *event* langsung didisposisi ke Wakil Dekan 2.
9. Wakil Dekan 2 menentukan jumlah dana yang akan dicairkan dan mengizinkan pengajuan peminjaman *venue*.
10. Jika dana yang disetujui kurang dari dana yang diajukan, maka sistem akan meminta konfirmasi pengaju *event* dalam hal ini unit internal fakultas. Jika menolak, maka unit internal fakultas dapat menemui Wakil Dekan 2 secara manual untuk berdiskusi lebih lanjut.

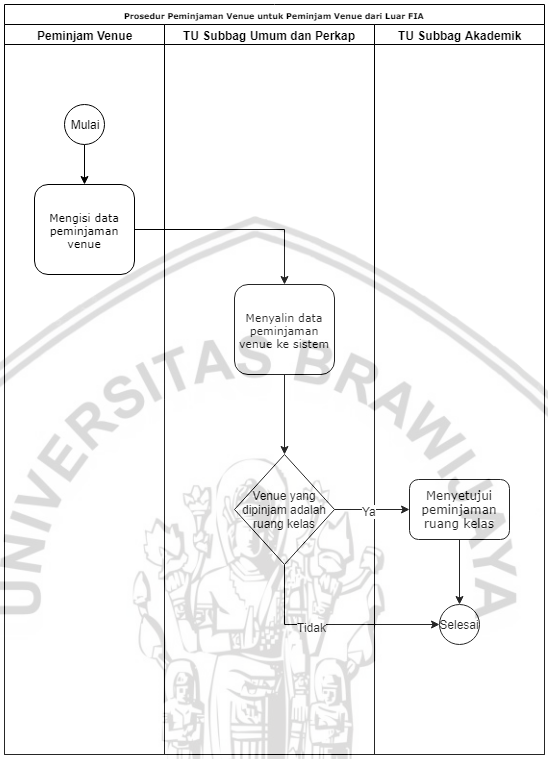


Gambar 4.6 BPM Prosedur Pengajuan Event Dekanat dan Unit Internal Fakultas

11. Setelah dana yang dicairkan disepakati, selanjutnya data pengajuan *event* dan proposal *event* akan didisposisi ke Kepala Tata Usaha.
12. Kepala Tata Usaha kemudian memerintahkan kepada unit-unit dibawahnya untuk menyediakan *resource* yang diminta oleh pengaju event. Jika *event* membutuhkan *venue* maka peminjaman *venue* memerlukan persetujuan TU Subbag Umum & Perlengkapan. Kemudian,

jika venue *event* merupakan ruang kelas, maka peminjaman *venue* juga memerlukan persetujuan dari TU Subbag Akademik.

13. Jika event membutuhkan dana, setelah LPJ/SPJ sudah diserahkan dan dana telah dicairkan maka TU Subbag Keuangan akan menginputkan data pencairan dana di sistem.



Gambar 4.7 BPM Prosedur Peminjaman *Venue*

Gambar 4.7 diatas merupakan prosedur peminjaman *venue* untuk peminjam *venue* yang merupakan organisasi/lembaga diluar FIA UB atau untuk peminjaman *venue* yang dapat dilakukan tanpa persetujuan Wakil Dekan 3 atau Wakil Dekan 2. Berikut ini merupakan penjelasan BPM prosedur peminjaman *venue* (Gambar 4.7):

1. Peminjam *venue* menemui *staff* TU Subbag Umum dan Perlengkapan.
2. *Staff* TU Subbag Umum dan Perlengkapan mengisi *form* peminjaman *venue* di sistem.
3. Jika *venue* tidak tersedia, maka peminjam memilih *venue* lain atau mengganti waktu peminjaman.
4. Jika *venue* yang dipinjam adalah ruang kelas, maka memerlukan persetujuan dari *staff* TU Subbag Akademik.

4.3 Identifikasi Aktor Sistem

Tabel 4.1 dibawah ini merupakan aktor yang teridentifikasi pada tahap analisis kebutuhan. Terdapat 16 aktor yang teridentifikasi dari hasil analisis kebutuhan.

Tabel 4.1 Identifikasi Aktor Sistem

Aktor	Deskripsi
Non Member	Merupakan aktor yang belum masuk kedalam sistem (hanya dapat melihat halaman Login).
Organisasi Mahasiswa	Merupakan aktor yang dapat mengajukan *event* yang berasal dari organisasi mahasiswa.
Unit Internal Fakultas	Merupakan aktor yang dapat mengajukan *event* yang berasal dari unit internal fakultas Misalnya jurusan, unit penunjang, unit tata usaha.
TU Subbag Kemahasiswaan	Merupakan aktor yang dapat mengajukan event. Selain itu TU Subbag Kemahasiswaan mempunyai *priviledge* tambahan yaitu dapat melihat seluruh event yang diajukan oleh organisasi mahasiswa.
Superadmin	Merupakan aktor yang berperan dalam mengelola data *event*, data *venue*, data akun dan data peminjaman *venue* di aplikasi ini.
Dekan	Merupakan aktor yang dapat mengajukan *event* (termasuk dalam pengaju event unit internal fakultas). Selain itu, Dekan juga berperan dalam menyetujui *event* jika penyelenggara event adalah unit internal fakultas atau organisasi mahasiswa (jika *event* akan diselenggarakan di luar kota).
Wakil Dekan 1	Merupakan aktor yang dapat mengajukan *event* (termasuk dalam pengaju *event* unit internal fakultas). Selain itu, Wakil Dekan 1 berperan dalam menyetujui *event* jika *event* yang diajukan merupakan *event* akademik.
Wakil Dekan 2	Merupakan aktor yang dapat mengajukan *event* (termasuk dalam pengaju *event* unit internal fakultas). Selain itu, Wakil Dekan 2 berperan dalam menyetujui *event*, mengizinkan pengajuan peminjaman venue dan menentukan jumlah dana yang akan dicairkan. Wakil Dekan 2 hanya menyetujui event yang penyelenggaranya adalah unit internal fakultas.
Wakil Dekan 3	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). Selain itu, Wakil Dekan 3 berperan dalam menyetujui event, mengizinkan pengajuan peminjaman venue dan menentukan jumlah dana yang akan dicairkan. Wakil Dekan 3 hanya menyetujui event

	yang penyelenggaranya adalah organisasi mahasiswa.
Kepala Tata Usaha	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). Aktor ini juga berperan dalam menyetujui pengajuan event apabila pengaju event merupakan unit internal yang membutuhkan *resource* dari unit-unit tata usaha. Kepala Tata Usaha juga berperan dalam menyetujui event apabila pengaju event adalah unit-unit tata usaha.
TU Subbag Keuangan	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). Aktor ini juga berperan dalam memasukkan data pencairan dana.
TU Subbag Umum dan Perlengkapan	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). TU Subbag Umum dan Perlengkapan juga dapat berperan didalam proses persetujuan peminjaman venue.
TU Subbag Akademik	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). TU Subbag Akademik dapat berperan didalam proses persetujuan peminjaman venue (jika venue yang dipinjam adalah ruang kelas).
Kepala Jurusan	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event unit internal fakultas). Kepala Jurusan juga berperan dalam menyetujui event dan menentukan jumlah dana yang akan dicairkan jika dana event berasal dari Jurusan atau jika pengaju event merupakan unit yang berada dibawah jurusan sesuai struktur organisasi. Masing-masing Jurusan di Fakultas Ilmu Administrasi Universitas Brawijaya memiliki satu akun.
BEM	Merupakan aktor yang dapat mengajukan event (termasuk dalam pengaju event organisasi mahasiswa). Selain itu BEM juga merupakan aktor yang berperan dalam menyetujui event.
Google Calendar	Aktor ini merupakan sistem eksternal yang akan digunakan oleh aplikasi Manajemen *Event* untuk menampilkan *event* dalam format kalender.

4.4 Definisi Kebutuhan Sistem

Definisi kebutuhan berisi seluruh definisi kebutuhan fungsional maupun non-fungsional yang telah teridentifikasi pada tahap analisis kebutuhan. Pada tahap analisis kebutuhan telah teridentifikasi pada aplikasi Manajemen *Event* terdapat 59 kebutuhan fungsional dan 1 kebutuhan non-fungsional. Sebagai bagian dari tahap manajemen kebutuhan yang bertujuan untuk mempermudah identifikasi

setiap kebutuhan, penulis telah menetapkan aturan penomoran kebutuhan sebagai berikut:

Gambar 4.8 Aturan Penomoran Kebutuhan

Keterangan:

AME : merupakan singkatan dari aplikasi ini yaitu Aplikasi Manajemen *Event.*

F/ NF : F menunjukkan bahwa kebutuhan merupakan kebutuhan fungsional. Sedangkan NF menunjukkan bahwa kebutuhan merupakan kebutuhan non-fungsional.

100 : No. urut kebutuhan, dimulai dari 100.

4.4.1 Definisi Kebutuhan Fungsional

Tabel 4.2 dibawah merupakan definisi kebutuhan fungsional yang teridentifikasi. Terdapat 59 definisi kebutuhan yang teridentifikasi pada tahap analisis kebutuhan.

Tabel 4.2 Definisi Kebutuhan Fungsional

No	Kode Kebutuhan	Definisi Kebutuhan	Nama Use Case
1	AME_F_100	Sistem harus dapat mengautentikasi dan mengauthorisasi aktor.	*Login*
2	AME_F_200	Sistem harus dapat mendeautentikasi aktor.	*Logout*
3	AME_F_300	Sistem harus dapat menampilkan halaman beranda.	Melihat beranda
4	AME_F_400	Sistem harus dapat menampilkan *list event* dari sisi pengaju event.	Melihat *list event* dari pengaju event
5	AME_F_500	Sistem harus dapat menampilkan halaman persetujuan peminjaman *venue*.	Melihat halaman persetujuan peminjaman *venue*
6	AME_F_600	Sistem harus dapat menampilkan halaman persetujuan *event*.	Melihat halaman persetujuan *event*
7	AME_F_700	Sistem harus dapat digunakan untuk memfilter event yang ditampilkan di list	Memfilter event di *list event*

		event.	
8	AME_F_800	Sistem harus dapat menampilkan halaman persetujuan *event* dari sisi WD2 dan WD3.	Melihat halaman persetujuan *event* dari WD2 dan WD3
9	AME_F_900	Sistem harus dapat menampilkan *detail* event.	Melihat detail *event*
10	AME_F_1000	Sistem harus dapat digunakan untuk mencari *event* dari list *event*.	Mencari *event*
11	AME_F_1100	Sistem harus dapat digunakan untuk mengajukan *event*.	Mengajukan *event*
12	AME_F_1200	Sistem harus dapat digunakan untuk mengedit data profil aktor.	Mengedit data profil
13	AME_F_1300	Sistem harus dapat digunakan untuk mengedit *password* aktor.	Mengedit *password*
14	AME_F_1400	Sistem harus dapat menampilkan jumlah notifikasi.	Melihat jumlah notifikasi
15	AME_F_1500	Sistem harus dapat digunakan untuk mengedit peminjaman *venue*.	Mengedit peminjaman venue
16	AME_F_1600	Sistem harus dapat digunakan untuk membatalkan persetujuan event.	Membatalkan persetujuan *event*
17	AME_F_1700	Sistem harus dapat digunakan untuk mengedit persetujuan event.	Mengedit persetujuan *event*
18	AME_F_1800	Sistem harus dapat digunakan untuk mengedit data pencairan dana.	Mengedit data pencairan dana
19	AME_F_1900	Sistem harus dapat digunakan untuk mengedit persetujuan *event* dari sisi *approver* dana.	Mengedit persetujuan *event* dari *approver* dana
20	AME_F_2000	Sistem harus dapat digunakan untuk menampilkan proposal *event*.	Melihat proposal *event*
21	AME_F_2100	Sistem harus dapat digunakan untuk menampilkan halaman notifikasi *event*.	Melihat halaman notifikasi *event*

22	AME_F_2200	Sistem harus dapat digunakan untuk menerima jumlah dana yang disetujui oleh *approver*.	Menerima jumlah dana yang disetujui
23	AME_F_2300	Sistem harus dapat digunakan untuk menolak jumlah dana yang disetujui oleh *approver*.	Menolak jumlah dana yang disetujui
24	AME_F_2400	Sistem harus dapat digunakan untuk merevisi proposal *event* yang ditolak oleh *approver*.	Merevisi proposal *event*
25	AME_F_2500	Sistem harus dapat menampilkan kalender *event*.	Melihat kalender *event*
26	AME_F_2600	Sistem harus dapat menampilkan *list event* dari sisi TU Subbag Kemahasiswaan.	Melihat list *event* dari TU Kemahasiswaan
27	AME_F_2700	Sistem harus dapat digunakan untuk memasukkan *event* kedalam Google Calendar.	Memasukkan *event* ke google calendar
28	AME_F_2800	Sistem harus dapat menampilkan *list event* dari sisi *approver* non dekanat.	Melihat *list event* dari *approver* non dekanat
29	AME_F_2900	Sistem harus dapat digunakan untuk mengedit proposal *event*.	Mengedit proposal
30	AME_F_3000	Sistem harus dapat digunakan untuk membatalkan *event* yang sebelumnya ditolak.	Membatalkan penolakan *event*
31	AME_F_3100	Sistem harus dapat digunakan untuk menyetujui *event*.	Menyetujui *event*
32	AME_F_3200	Sistem harus dapat digunakan untuk membatalkan pengajuan *event*	Membatalkan pengajuan *event*
33	AME_F_3300	Sistem harus dapat digunakan untuk menyetujui *event* dari sisi WD2 dan WD3.	Menyetujui *event* dari WD2 dan WD3
34	AME_F_3400	Sistem harus dapat digunakan untuk menolak *event* yang diajukan.	Menolak *event*
35	AME_F_3500	Sistem harus dapat digunakan untuk memasukan data pencairan dana.	Memasukkan data pencairan dana

36	AME_F_3600	Sistem harus dapat digunakan untuk menyetujui peminjaman *venue* dari sisi TU Subbag Umum dan Perlengkapan.	Menyetujui peminjaman *venue* dari TU Umum Perkap
37	AME_F_3700	Sistem harus dapat digunakan untuk memasukkan data peminjaman *venue*.	Memasukkan peminjaman *venue*
38	AME_F_3800	Sistem harus dapat menampilkan seluruh *list event*.	Melihat seluruh *list event*
39	AME_F_3900	Sistem harus dapat digunakan untuk menghapus *event* di Google Calendar.	Menghapus *event* di Google Calendar
40	AME_F_4000	Sistem harus dapat menampilkan list *venue*	Melihat *list venue*
41	AME_F_4100	Sistem harus dapat digunakan untuk memasukkan *venue* baru	Memasukkan *venue* baru
42	AME_F_4200	Sistem harus dapat digunakan untuk memasukkan *venue* menggunakan file CSV	Memasukkan *venue* menggunakan CSV
43	AME_F_4300	Sistem harus dapat digunakan untuk menghapus *venue* dari *list venue*	Menghapus *venue*
44	AME_F_4400	Sistem harus dapat digunakan untuk mengedit *venue*.	Mengedit *venue*
45	AME_F_4500	Sistem harus dapat digunakan untuk mencari *venue* dari *list venue*.	Mencari *venue*
46	AME_F_4600	Sistem harus dapat digunakan untuk mengedit *event* di Google Calendar.	Mengedit *event* di Google Calendar
47	AME_F_4700	Sistem harus dapat menampilkan *list* peminjaman *venue* dari TU Akademik.	Melihat *list* peminjaman *venue* dari TU Akademik
48	AME_F_4800	Sistem harus dapat digunakan untuk membuat akun.	Membuat akun baru
49	AME_F_4900	Sistem harus dapat menampilkan *list* akun.	Melihat *list* akun
50	AME_F_5000	Sistem harus dapat digunakan untuk mencari akun dari *list* akun.	Mencari akun

51	AME_F_5100	Sistem harus dapat digunakan untuk mengedit akun.	Mengedit akun
52	AME_F_5200	Sistem harus dapat digunakan untuk menghapus akun.	Menghapus akun
53	AME_F_5300	Sistem harus dapat digunakan untuk mereset *password*.	Mereset *password*
54	AME_F_5400	Sistem harus dapat menampilkan seluruh *list* peminjaman *venue.*	Melihat seluruh *list* peminjaman venue
55	AME_F_5500	Sistem harus dapat digunakan untuk menghapus peminjaman *venue.*	Menghapus peminjaman venue
56	AME_F_5600	Sistem harus dapat digunakan untuk menghapus *event*.	Menghapus *event*
57	AME_F_5700	Sistem harus dapat digunakan untuk mengedit data *event*.	Mengedit data *event*
58	AME_F_5800	Sistem harus dapat digunakan untuk menyetujui peminjaman *venue* dari sisi TU Subbag Akademik.	Menyetujui peminjaman venue dari TU Akademik
59	AME_F_5900	Sistem harus dapat menampilkan detail akun.	Menampilkan detail akun

4.4.2 Definisi Kebutuhan Non Fungsional

Tabel 4.3 dibawah ini merupakan definisis kebutuhan non-fungsional dari penelitian ini. Terdapat satu kebutuhan non-fungsional yang teridentifikasi pada tahap analisis kebutuhan.

Tabel 4.3 Definisi Kebutuhan Non Fungsional

No	Kode	Parameter	Deskripsi
1	AME_NF_100	*Browser Compatibility Testing*	Sistem harus dapat dijalankan secara sempurna (tanpa kehilangan fungsionalitas) di berbagai web *browser* yaitu Google Chrome, Microsoft Edge, Firefox, Opera dan Safari.

4.5 Spesifikasi Kebutuhan Sistem

Spesifikasi kebutuhan sistem berisi pendetailan dari seluruh kebutuhan yang telah didefinisikan sebelumnya pada tahap definisi kebutuhan. Pada spesifikasi

kebutuhan, penomoran tiap spesifikasi kebutuhan disesuaikan dengan nomor definisi kebutuhan yang dijelaskannya.

4.5.1 Spesifikasi Kebutuhan Fungsional

Tabel 4.4 dibawah ini merupakan spesifikasi kebutuhan fungsional yang teridentifikasi pada tahap analisis kebutuhan. Spesifikasi kebutuhan merupakan pendetailan dari definisi kebutuhan yang telah dijelaskan pada subbab 4.4.1.

Tabel 4.4 Spesifikasi Kebutuhan Fungsional

No	Kode Definisi Kebutuhan	Kode Spesifikasi Kebutuhan	Spesifikasi Kebutuhan	Aktor
1	AME_F_100	AME_F_101	Sistem harus menyediakan *form* yang berisi *field* NIP/NIM dan *password.*	Non Member
		AME_F_102	Aktor dapat masuk ke dalam sistem melalui dua atau lebih perangkat yang berbeda dalam satu waktu.	Seluruh Aktor kecuali Non Member
		AME_F_103	Session akan *expired* dalam waktu 1 x 24 jam.	Seluruh Aktor kecuali Non Member
		AME_F_104	Aktor harus melakukan login ulang apabila *session* telah *expired.*	Seluruh Aktor kecuali Non Member
		AME_F_105	*Field* NIM/NIP tidak boleh dikosongkan	Non Member
		AME_F_106	Input pada *field* NIM/NIP dibatasi minimal 10 karakter dan maksimal 30 karakter.	Non Member
		AME_F_107	Input pada field password dibatasi minimal 8 karakter dan maksimal 100 karakter.	Non Member
2	AME_F_200	AME_F_201	Sistem harus menyediakan *button* untuk keluar dari sistem.	Seluruh aktor kecuali non member
3	AME_F_300	AME_F_301	Pada halaman beranda terdapat 3 *button* yaitu	Seluruh aktor kecuali non

			button yang mengarahkan aktor ke halaman pengajuan event, *button* yang mengarahkan aktor ke halaman kalender *event* dan *button* yang menampilkan *modal* yang berisi langkah-langkah untuk menyalin kalender *event*.	member
4	AME_F_400	AME_F_401	*Event* yang ditampilkan pada *list* ini hanyalah *event* yang pernah diajukan oleh aktor.	Organisasi Mahasiswa, Unit Internal Fakultas
		AME_F_402	Sistem harus dapat digunakan untuk menentukan jumlah *list event* yang ditampilkan perhalamannya yang diimplementasikan menggunakan *library* DataTables.	Organisasi Mahasiswa, Unit Internal Fakultas
		AME_F_403	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	Organisasi Mahasiswa, Unit Internal Fakultas
5	AME_F_500	AME_F_501	Halaman persetujuan venue berisi *list event* yang perlu mendapatkan persetujuan dari aktor untuk dapat menggunakan *venue*.	TU Subbag Umum & Perlengkapan, TU Subbag Akademik
		AME_F_502	Didalam setiap *collapsible* terdapat *form* catatan dan *button* untuk mengirimkan *form* persetujuan peminjaman venue.	TU Subbag Umum & Perlengkapan, TU Subbag Akademik
		AME_F_503	*Field* catatan tidak boleh	TU Subbag Umum &

			dikosongkan.	Perlengkapan, TU Subbag Akademik
6	AME_F_600	AME_F_601	Halaman persetujuan *event* berisi *list event* yang perlu mendapatkan persetujuan dari *approver* yang hanya menyetujui proposal event.	Dekan, WD1, BEM, KTU, Kajur
		AME_F_602	Didalam setiap *collapsible* terdapat *form* catatan dan *button* untuk mengirimkan *form* persetujuan *event*.	Dekan, WD1, BEM, KTU, Kajur
		AME_F_603	*Field* catatan tidak boleh dikosongkan.	Dekan, WD1, BEM, KTU, Kajur
7	AME_F_700	AME_F_701	Filter harus diimplementasikan dalam bentuk *dropdown.*	TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
		AME_F_702	Kategori yang terdapat pada filter adalah Tampilkan Hanya Event Saya dan Tampilkan Seluruh Event	TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
		AME_F_703	Jika aktor memilih kategori Tampilkan	TU Subbag Kemahasiswaan,

			Hanya Event Saya, maka sistem akan menampilkan event yang pernah diajukan aktor tersebut.	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
		AME_F_704	Jika aktor memilih kategori Tampilkan Seluruh Event maka sistem akan menampilkan seluruh event yang ada.	TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
8	AME_F_800	AME_F_801	Jika dana berasal dari Fakultas/Kemahasiswaan, maka sistem harus menampilkan *field* jumlah dana yang disetujui pada halaman persetujuan ini.	WD2, WD3
		AME_F_802	Jika event membutuhkan *venue*, maka sistem harus menampilkan *checkbox* untuk mengizinkan pengajuan peminjaman *venue*.	WD2, WD3
		AME_F_803	Didalam setiap *collapsible* terdapat form catatan dan button untuk mengirimkan form persetujuan *event*.	WD2, WD3
		AME_F_804	*Field* catatan tidak boleh dikosongkan.	WD2, WD3

9	AME_F_900	AME_F_901	Halaman detail event yang seluruh informasi terkait *event*.	Seluruh Aktor kecuali Non Member
		AME_F_902	Didalam *field* status pengajuan event, sistem harus menyediakan informasi terkait status persetujuan event (sudah disetujui atau belum).	Seluruh Aktor kecuali Non Member
		AME_F_903	Didalam *field* status pengajuan event juga terdapat nama *approver*, NIP/NIM *approver* dan tanggal serta waktu event disetujui.	Seluruh Aktor kecuali Non Member
		AME_F_904	Aktor hanya dapat melihat detail *event* dari *event* yang berada di halaman *list event* miliknya.	Seluruh Aktor kecuali Non Member
10	AME_F_1000	AME_F_1001	Fungsionalitas untuk mencari *venue* yang diimplementasikan menggunakan *library* DataTables.	Seluruh Aktor kecuali Non Member
11	AME_F_1100	AME_F_1101	Form pengajuan event terdiri dari field Penyelenggara, Judul Kegiatan, Jenis *Event*, Deskripsi *Event*, File Proposal, Asal Dana, Jumlah Dana yang Diajukan	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1102	Pada halaman aktor yang bukan merupakan organisasi mahasiswa, sistem harus menampilkan *field* untuk memilih jenis *event.*	Seluruh Aktor kecuali Non Member, Organisasi Mahasiswa, BEM dan Super Admin
		AME_F_1103	Selain itu, pada form pengajuan, sistem juga	Seluruh Aktor kecuali Non

			harus menyediakan 3 tab yaitu peminjaman venue untuk *event* tidak rutin, *event* rutin dan tidak membutuhkan *venue*.	Member dan Super Admin
		AME_F_1104	Tab peminjaman venue untuk *event* tidak rutin digunakan aktor untuk menginputkan data peminjaman *venue* untuk *event* yang bukan merupakan *event* rutin.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1105	Pada tab ini, sistem harus menyediakan *field* nama lengkap peminjam, NIP/NIM, jumlah personel yang terlibat, venue yang dipinjam (aktor dapat memilih lebih dari 1 venue), tanggal peminjaman, waktu mulai dan selesai, Lain lain dan checkbox untuk menyetujui syarat dan ketentuan peminjaman venue di FIA UB	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1106	Tab peminjaman venue untuk *event* tidak rutin digunakan aktor untuk menginputkan data peminjaman venue untuk event yang merupakan event rutin.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1107	Pada tab ini, sistem harus menyediakan field nama lengkap peminjam, NIP/NIM, jumlah personel yang terlibat, *checkbox* untuk memilih hari (senin-minggu), waktu mulai, waktu selesai, tanggal mulai jadwal rutin, tanggal	Seluruh Aktor kecuali Non Member dan Super Admin

			selesai jadwal rutin, venue yang dipinjam (aktor dapat memilih lebih dari 1 venue), lain-lain dan *checkbox* untuk menyetujui syarat dan ketentuan peminjaman venue di FIA UB	
		AME_F_1108	*Tab* peminjaman venue untuk *event* tidak rutin digunakan jika *event* tidak membutuhkan *venue* di FIA.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1109	Pada tab ini, sistem harus menyediakan field tanggal mulai, tanggal selesai dan *radio button* sebagai sarana aktor untuk menginputkan apakah event diakan didalam kota Malang atau diluar kota Malang	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1110	Sistem juga harus menyediakan mekanisme pengecekan ketersediaan *venue* ketika aktor mengajukan *event*	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1111	Aktor hanya dapat mengisi salah satu dari tiga tab yang tersedia di data peminjaman venue yaitu tab peminjaman venue untuk e*vent* tidak rutin, untuk e*vent* rutin dan tidak memerlukan v*enue*.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1112	Input pada field nama lengkap pengaju, NIP/NIM, no. telepon, email pengaju, jumlah personel dibatasi minimal 5 karakter dan maksimal 100 karakter.	Seluruh Aktor kecuali Non Member dan Super Admin

		AME_F_1113	Input pada field judul kegiatan dan deskripsi event dibatasi minimal 10 karakter dan maksimal 500 karakter.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1114	Input pada field file proposal hanya menerima file berekstensi .pdf.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1115	Input pada *field venue* yang dipinjam harus berupa *dropdown* yang berisi seluruh nama venue yang berasal dari database.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1116	*Field* asal dana didisabled dan berisi nilai "Fakultas"	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1117	Nilai valid untuk dropdown jenis event adalah "akademik" dan "non akademik"	Seluruh Aktor kecuali Non Member, Organisasi Mahasiswa, BEM dan Super Admin
		AME_F_1118	*Field* jumlah dana hanya menerima format *number*.	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1119	Input pada field tanggal mulai dan tanggal selesai harus bertipe *date*	Seluruh Aktor kecuali Non Member dan Super Admin
		AME_F_1120	Input pada field waktu mulai dan waktu selesai harus bertipe time	Seluruh Aktor kecuali Non Member dan Super Admin
12	AME_F_1200	AME_F_1201	Sistem harus menyediakan form yang berisi field NIP/NIM, nama organisasi, nama	Seluruh Aktor kecuali Non Member

			ketua organisasi, email dan no.hp.	
		AME_F_1202	Input pada field NIP/NIM, dibatasi minimal 10 karakter dan maksimal 30 karakter dan harus berformat number.	Seluruh Aktor kecuali Non Member
		AME_F_1203	Input pada field nama organisasi dan nama ketua organisasi dibatasi minimal 10 karakter dan maksimal 100 karakter.	Seluruh Aktor kecuali Non Member
		AME_F_1204	Input pada field No HP dibatasi minimal 10 karakter dan maksimal 13 karakter dan harus berformat number.	Seluruh Aktor kecuali Non Member
		AME_F_1205	Seluruh field pada form edit data profil tidak boleh dikosongkan	Seluruh Aktor kecuali Non Member
13	AME_F_1300	AME_F_1301	Sistem harus menyediakan form yang berisi field password lama, password baru dan konfirmasi password baru.	Seluruh Aktor kecuali Non Member
		AME_F_1302	Nilai yang diinputkan aktor pada field password baru dan konfirmasi password baru harus sama.	Seluruh Aktor kecuali Non Member
		AME_F_1303	Input seluruh field pada form edit password dibatasi minimal 10 karakter dan maksimal 50 karakter.	Seluruh Aktor kecuali Non Member
		AME_F_1304	Seluruh field pada form edit password tidak boleh dikosongkan	Seluruh Aktor kecuali Non Member
14	AME_F_1400	AME_F_1401	Jumlah notifikasi ditampilkan dalam	Seluruh Aktor kecuali Non

			bentuk *badge* yang berada di *sidebar* dan tab pada halaman notifikasi.	Member dan Super Admin
		AME_F_1402	Jumlah notifikasi sama dengan jumlah notifikasi yang ada pada halaman melihat notifikasi.	Seluruh Aktor kecuali Non Member dan Super Admin
15	AME_F_1500	AME_F_1501	Sistem harus menyediakan form yang berisi nama venue yang dipinjam, tanggal peminjaman venue dan waktu mulai serta waktu selesai	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_1502	Field nama venue merupakan dropdown yang berisi seluruh nama venue yang ada di database	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_1503	Field tanggal peminjaman venue harus berformat date	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_1504	Field waktu mulai dan selesai harus berformat time	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_1505	Seluruh field pada form edit peminjaman venue tidak boleh kosong	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
16	AME_F_1600	AME_F_1601	Jika tombol tersebut diklik maka sistem harus menampilkan alert konfirmasi.	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM

		AME_F_1602	Aktor hanya dapat membatalkan event yang sebelumnya telah ia setujui.	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM
17	AME_F_1700	AME_F_1701	Sistem harus menyediakan form yang berisi field catatan dan button untuk mengirimkan form tersebut	WD1, KTU, BEM, Dekan, Kajur
		AME_F_1702	Field catatan tidak boleh dikosongkan	WD1, KTU, BEM, Dekan, Kajur
18	AME_F_1800	AME_F_1801	Sistem harus menyediakan form yang berisi field nama pengambil dana, nip/nim pengambil dana, waktu pengambilan dana dan button untuk mengirimkan form tersebut.	TU Subbag Keuangan
		AME_F_1802	Tidak boleh ada field pada form edit pencairan dana yang dikosongkan	TU Subbag Keuangan
		AME_F_1803	Field NIP/NIM pengambil dana hanya menerima inputan angka dan maksimal terdiri dari 30 karakter.	TU Subbag Keuangan
		AME_F_1804	Field nama pengambil dana minimal terdiri dari 5 karakter dan maksimal 100 karakter.	TU Subbag Keuangan
		AME_F_1805	Field tanggal pengambilan dana harus bertipe DATETIME	TU Subbag Keuangan
19	AME_F_1900	AME_F_1901	Fungsionalitas ini hanya dapat diakses oleh approver yang sebelumnya telah menyetujui jumlah dana	WD2, WD3

			yang dicairkan.	
		AME_F_1902	Sistem harus menyediakan form yang berisi field jumlah dana yang disetujui dan catatan serta *button* untuk mengirimkan *form* tersebut.	WD2, WD3
		AME_F_1903	Tidak boleh ada field pada form edit persetujuan dana yang dikosongkan	WD2, WD3
		AME_F_1904	Field jumlah dana yang disetujui hanya menerima input berupa angka	WD2, WD3
		AME_F_1905	Field catatan maksimal terdiri dari 500 karakter	WD2, WD3
20	AME_F_2000	AME_F_2001	Sistem harus mengarahkan aktor ke *tab* lain apabila aktor mengklik link proposal.	Seluruh Aktor kecuali Non Member
21	AME_F_2100	AME_F_2101	Halaman notifikasi event berisi field untuk merevisi proposal apabila suatu event ditolak oleh approver	Seluruh aktor kecuali Non Member dan Super Admin
		AME_F_2102	Selain itu, halaman notifikasi berisi field untuk mengkonfirmasi dan menolak dana jika jumlah dana yang disetujui oleh approver kurang dari jumlah dana yang diajukan oleh pengaju event	Seluruh aktor kecuali Non Member dan Super Admin
22	AME_F_2200	AME_F_2201	Sistem harus menyediakan button untuk menerima jumlah dana yang disetujui oleh *approver*	Seluruh aktor kecuali Non Member dan Super Admin

23	AME_F_2300	AME_F_2301	Sistem harus menampilkan *alert* konfirmasi apabila aktor menolak dana	Seluruh aktor kecuali Non Member dan Super Admin
24	AME_F_2400	AME_F_2401	Sistem harus menyediakan form yang berisi field untuk mengupload file proposal	Seluruh aktor kecuali Non Member dan Super Admin
		AME_F_2402	Field hanya menerima file proposal yang berformat .pdf	Seluruh aktor kecuali Non Member dan Super Admin
		AME_F_2403	Field untuk meng*upload* file proposal tidak boleh dikosongkan	Seluruh aktor kecuali Non Member dan Super Admin
25	AME_F_2500	AME_F_2501	Sistem harus menampilkan kalender yang berasal dari aplikasi Google Calendar	Seluruh aktor kecuali Non Member
26	AME_F_2600	AME_F_2601	Event yang ditampilkan pada halaman ini adalah event yang pernah diajukan TU Subbag Kemahasiswaan dan seluruh event yang pernah diajukan oleh organisasi mahasiswa	TU Subbag Kemahasiswaan
		AME_F_2602	Sistem harus dapat digunakan untuk menentukan jumlah *list event* yang ditampilkan perhalamannya yang diimplementasikan menggunakan *library* DataTables.	TU Subbag Kemahasiswaan
		AME_F_2603	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	TU Subbag Kemahasiswaan

27	AME_F_2700	AME_F_2701	Sistem harus dapat terhubung dengan aplikasi Google Calendar melalui Google Calendar API.	TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
28	AME_F_2800	AME_F_2801	Event yang ditampilkan pada halaman ini adalah event yang pernah diajukan oleh aktor dan seluruh event yang pernah mendapatkan persetujuan dari aktor.	Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
		AME_F_2802	Sistem harus dapat digunakan untuk menentukan jumlah list event yang ditampilkan perhalamannya yang diimplementasikan menggunakan library DataTables.	Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
		AME_F_2803	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
29	AME_F_2900	AME_F_2901	Sistem harus menyediakan form yang berisi field untuk mengupload file proposal	Super Admin
		AME_F_2902	Field proposal tidak boleh dikosongkan dan field hanya menerima file dengan format .pdf	Super Admin
30	AME_F_3000	AME_F_3001	Aktor hanya dapat membatalkan penolakan suatu event yang sebelumnya telah ditolak oleh aktor tersebut	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM

31	AME_F_3100	AME_F_3101	Sistem harus menyediakan form berisi field catatan dan button untuk mengirimkan form.	Dekan, WD1, KTU, BEM, Kajur
		AME_F_3102	Field catatan tidak boleh dikosongkan dan maksimal terdiri dari 500 karakter.	Dekan, WD1, KTU, BEM, Kajur
32	AME_F_3200	AME_F_3201	Sistem harus menyediakan *button* untuk membatalkan pengajuan *event*.	Seluruh aktor kecuali non member dan super admin
33	AME_F_3300	AME_F_3301	Jika aktor mengklik tombol setujui maka sistem harus dapat memastikan semua field pada form persetujuan event telah diisi oleh aktor	WD2, WD3
34	AME_F_3400	AME_F_3401	Sistem harus menampilkan *alert* konfirmasi jika aktor menolak *event*	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM
35	AME_F_3500	AME_F_3501	Sistem harus menyediakan form yang berisi checkbox verifikasi SPJ/LPJ, NIM/NIP pengambil dana dan nama pengambil dana.	TU Subbag Keuangan
		AME_F_3502	Semua field yang ada pada form pencairan dana tidak boleh dikosongkan.	TU Subbag Keuangan
		AME_F_3503	Field NIP/NIM pengambil dana hanya menerima inputan angka dan maksimal terdiri dari 30 karakter.	TU Subbag Keuangan
		AME_F_3504	Field nama pengambil dana minimal terdiri dari 5 karakter dan maksimal	TU Subbag Keuangan

			100 karakter.	
36	AME_F_3600	AME_F_3601	Sistem harus menyediakan form yang berisi field catatan dan *button* setujui untuk menyetujui peminjaman venue.	TU Subbag Umum dan Perlengkapan
		AME_F_3601	Field catatan tidak boleh dikosongkan dan maksimal terdiri dari 500 karakter.	TU Subbag Umum dan Perlengkapan
37	AME_F_3700	AME_F_3701	Form pengajuan event terdiri dari field Penyelenggara, Judul Kegiatan, Jenis Event, Deskripsi Event, File Proposal, Asal Dana, Jumlah Dana yang Diajukan	TU Subbag Umum dan Perlengkapan
		AME_F_3702	Selain itu, pada form pengajuan, sistem juga harus menyediakan tab untuk memasukkan data peminjaman venue	TU Subbag Umum dan Perlengkapan
		AME_F_3703	Pada tab ini, sistem harus menyediakan field nama lengkap peminjam, NIP/NIM, jumlah personel yang terlibat, venue yang dipinjam (aktor dapat memilih lebih dari 1 venue), tanggal peminjaman, waktu mulai dan selesai, Lain lain dan checkbox untuk menyetujui syarat dan ketentuan peminjaman venue di FIA UB	TU Subbag Umum dan Perlengkapan
		AME_F_3704	Sistem juga harus menyediakan mekanisme pengecekan ketersediaan venue ketika aktor	TU Subbag Umum dan Perlengkapan,

			mengajukan event	Super Admin
		AME_F_3705	Sistem harus dapat memastikan bahwa semua field kecuali field "Lain-lain" telah diisi oleh aktor.	TU Subbag Umum dan Perlengkapan
		AME_F_3706	Input pada field nama lengkap pengaju, NIP/NIM, no. telepon, email pengaju, jumlah personel dibatasi minimal 5 karakter dan maksimal 100 karakter.	TU Subbag Umum dan Perlengkapan
		AME_F_3707	Input pada field judul kegiatan dan deskripsi event dibatasi minimal 10 karakter dan maksimal 500 karakter.	TU Subbag Umum dan Perlengkapan
		AME_F_3708	Input pada field file proposal hanya menerima file berekstensi .pdf.	TU Subbag Umum dan Perlengkapan
		AME_F_3709	Input pada field venue yang dipinjam harus berupa *dropdown*.	TU Subbag Umum dan Perlengkapan
		AME_F_3710	Input pada field tanggal mulai dan tanggal selesai harus bertipe *date*	TU Subbag Umum dan Perlengkapan
		AME_F_3711	Input pada field waktu mulai dan waktu selesai harus bertipe time	TU Subbag Umum dan Perlengkapan
38	AME_F_3800	AME_F_3800	Sistem harus dapat digunakan untuk menentukan jumlah list event yang ditampilkan perhalamannya yang diimplementasikan menggunakan library DataTables.	Superadmin, Dekan, WD1, WD2, WD3
		AME_F_3801	Sistem harus menyediakan sarana	Superadmin, Dekan, WD1,

			berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	WD2, WD3
39	AME_F_3900	AME_F_3901	Sistem harus dapat menghapus event di Google Calendar menggunakan Google Calendar API	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
40	AME_F_4000	AME_F_4001	Sistem harus menyediakan sarana untuk menampilkan daftar *venue* berupa tabel.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4002	Tabel tersebut berisi kolom no, nama *venue*, jenis *venue* dan action.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4003	Kolom action berisi tombol hapus dan edit.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4005	Sistem harus dapat digunakan untuk menentukan jumlah list venue yang ditampilkan perhalamannya yang diimplementasikan menggunakan library DataTables.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4006	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik

41	AME_F_4100	AME_F_4101	Form tersebut berisi field nama venue dan keterangan.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4102	Tidak boleh ada field pada form tambah venue yang dikosongkan	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4103	Field nama venue minimal terdiri dari 5 karakter dan maksimal 100 karakter	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4104	Sistem harus menyediakan field untuk memasukkan keterangan dalam bentuk *dropdown.*	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4105	Pilihan yang valid untuk *dropdown* jenis venue adalah ruang kelas dan bukan ruang kelas.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
42	AME_F_4200	AME_F_4201	File yang diupload haruslah file yang berformat CSV	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4202	Field file upload file CSV tidak boleh dikosongkan	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik

		AME_F_4203	Nama kolom dan jumlah kolom yang terdapat pada file CSV harus sesuai dengan kolom yang terdapat di database	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
43	AME_F_4300	AME_F_4301	Jika aktor mengklik tombol hapus maka sistem harus dapat menampilkan alert konfirmasi.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
44	AME_F_4400	AME_F_4401	Sistem harus menyediakan sarana berupa form yang berisi nama venue dan jenis venue serta tombol untuk mengedit venue.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4402	Seluruh field pada form edit venue tidak boleh dikosongkan.	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4403	Field nama venue merupakan dropdown dan data didalam dropdown merupakan seluruh nama venue yang ada di database	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
		AME_F_4404	Field keterangan merupakan dropdown dimana nilai yang valid adalah ruang kelas dan bukan ruang kelas	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
45	AME_F_4500	AME_F_4501	Fungsionalitas untuk mencari venue yang diimplementasikan menggunakan *library* DataTables.	TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, Super Admin

46	AME_F_4600	AME_F_4601	Sistem harus dapat mengedit event di Google Calendar menggunakan Google Calendar API	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
47	AME_F_4700	AME_F_4701	Sistem harus menampilkan peminjaman venue dimana venue yang dipinjam adalah ruang kelas	TU Subbag Akademik
		AME_F_4702	Sistem harus dapat digunakan untuk menentukan jumlah list peminjaman venue yang ditampilkan perhalamannya yang diimplementasikan menggunakan *library* DataTables.	TU Subbag Akademik
		AME_F_4703	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables..	TU Subbag Akademik
48	AME_F_4800	AME_F_4801	Form tersebut berisi field NIP/NIM, nama organisasi, nama ketua organisasi, email, no.telepon, role, keterangan	Super Admin
		AME_F_4802	Seluruh field pada form tambah akun tidak boleh dikosongkan	Super Admin
		AME_F_4803	Input pada field NIP/NIM, nama organisasi, nama ketua organisasi, email, no.telepon dibatasi minimal 8 karakter dan maksimal 100 karakter	Super Admin

		AME_F_4804	Field no telepon hanya menerima inputan berupa angka dan minimal terdiri dari 10 karakter dan maksimal 13 karakter.	Super Admin
		AME_F_4805	Field email pengaju harus bertipe email	Super Admin
		AME_F_4806	Field NIP/NIM minimal terdiri dari 10 karakter dan maksimal 30 karakter	Super Admin
		AME_F_4807	Field nama ketua organisasi dan nama organisasi serta email minimal terdiri dari 10 karakter dan maksimal 100 karakter	Super Admin
		AME_F_4808	Field role dan keterangan harus diimplementasikan dalam bentuk dropdown	Super Admin
		AME_F_4809	Nilai valid pada dropdown role adalah Pengaju event (mahasiswa), pengaju event unit internal fakultas, dekan, WD1, WD2, WD3, Kajur, KTU, TU Akademik, TU Kemahasiswaan, TU Umum dan Perlengkapan, TU Keuangan dan BEM	Super Admin
		AME_F_4810	Nilai valid pada dropodown keterangan adalah organisasi mahasiswa, dekanat, dibawah KTU, tidak ada, dibawah jurusan bisnis dan dibawah jurusan publik	Super Admin
		AME_F_4811	Sistem harus mengirimkan email ke	Super Admin

			email pembuat akun yang berisi password untuk masuk kedalam akun	
49	AME_F_4900	AME_F_4901	Sistem harus menyediakan sarana untuk menampilkan list akun berupa tabel.	Super Admin
		AME_F_4902	Tabel tersebut berisi kolom NIP/NIM, nama organisasi, nama ketua organisasi, email, no.telepon, role, keterangan dan action.	Super Admin
		AME_F_4903	Kolom action berisi tombol hapus dan edit serta reset.	Super Admin
		AME_F_4904	Sistem harus dapat digunakan menentukan jumlah list akun yang ditampilkan perhalamannya yang diimplementasikan menggunakan library DataTables.	Super Admin
		AME_F_4905	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	Super Admin
50	AME_F_5000	AME_F_5001	Fungsionalitas untuk mencari akun yang diimplementasikan menggunakan *library* DataTables.	Super Admin
51	AME_F_5100	AME_F_5101	Di halaman edit terdapat form yang berisi field NIP/NIM, nama organisasi, nama ketua organisasi, email, no.telepon, role,	Super Admin

			keterangan serta tombol untuk mengirimkan form tersebut.	
		AME_F_5102	Seluruh field pada form edit akun tidak boleh dikosongkan	Super Admin
		AME_F_5103	Field no telepon dan NIP/NIM hanya menerima format number	Super Admin
		AME_F_5104	Field email hanya menerima format email	Super Admin
		AME_F_5105	Field role dan keterangan harus diimplementasikan dalam bentuk dropdown	Super Admin
		AME_F_5106	Nilai valid pada dropdown role adalah Pengaju event (mahasiswa), pengaju event unit internal fakultas, dekan, WD1, WD2, WD3, Kajur, KTU, TU Akademik, TU Kemahasiswaan, TU Umum dan Perlengkapan, TU Keuangan dan BEM	Super Admin
		AME_F_5107	Nilai valid pada dropodown keterangan adalah organisasi mahasiswa, dekanat, dibawah KTU, tidak ada, dibawah jurusan bisnis dan dibawah jurusan publik	Super Admin
		AME_F_5108	Field NIP/NIM minimal terdiri dari 10 karakter dan maksimal 30 karakter	Super Admin
		AME_F_5109	Field nama ketua organisasi dan nama organisasi serta email minimal terdiri dari 10	Super Admin

			karakter dan maksimal 100 karakter	
		AME_F_5110	Field no hp minimal terdiri dari 10 karakter dan maksimal 13 karakter	Super Admin
52	AME_F_5200	AME_F_5201	Jika aktor mengklik *button* hapus maka sistem harus dapat menampilkan *alert* konfirmasi.	Super Admin
53	AME_F_5300	AME_F_5301	Sistem harus mengirimkan email yang berisi password baru yang di*generate* secara otomatis oleh sistem	Super Admin
		AME_F_5302	Password yang digenerate sistem terdiri dari huruf kecil acak sepanjang 10 karakter	Super Admin
54	AME_F_5400	AME_F_5401	Sistem harus dapat digunakan untuk menentukan jumlah list peminjaman venue yang ditampilkan perhalamannya yang diimplementasikan menggunakan *library* DataTables.	Super Admin, TU Subbag Umum dan Perlengkapan
		AME_F_5402	Sistem harus menyediakan sarana berupa *pagination* yang diimplementasikan menggunakan *library* DataTables.	Super Admin, TU Subbag Umum dan Perlengkapan
55	AME_F_5500	AME_F_5501	Jika peminjaman venue sebelumnya telah dimasukkan kedalam Google Calendar, maka sistem juga harus menghapus data peminjaman venue dari Google Calendar.	TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, Super Admin

56	AME_F_5600	AME_F_5601	Jika aktor mengklik tombol hapus maka sistem harus dapat menampilkan alert konfirmasi.	Super Admin
		AME_F_5602	Jika event sebelumnya telah dimasukkan ke Google Calendar, maka sistem juga harus menghapus data event dari Google Calendar.	Super Admin
57	AME_F_5700	AME_F_5701	Sistem harus menyediakan form yang berisi field judul kegiatan, deskripsi event dan jumlah dana yang diajukan dan button yang digunakan untuk mengedit event	Super Admin
		AME_F_5702	Seluruh field tidak boleh dikosongkan oleh aktor	Super Admin
		AME_F_5703	Field judul kegiatan minimal terdiri dari 20 karakter dan maksimal 250 karakter	Super Admin
		AME_F_5704	Field deskripsi event maksimal terdiri dari 500 karakter	Super Admin
		AME_F_5705	Field jumlah dana yang diajukan hanya menerima input angka	Super Admin
58	AME_F_5800	AME_F_5801	Sistem harus menyediakan form yang berisi field catatan dan *button* setujui untuk menyetujui peminjaman venue.	TU Subbag Akademik
		AME_F_5802	Jika peminjaman venue yang disetujui merupakan peminjaman venue yang diinputkan oleh TU Subbag Umum	TU Subbag Akademik

			dan Perlengkapan melalui halaman Peminjaman Venue, maka sistem harus memasukkan event ke Google Calendar	
59	AME_F_5900	AME_F_5901	Pada halaman detail akun terdapat informasi akun yang ditampilkan didalam tabel.	Super Admin
		AME_F_5902	Informasi akun terdiri dari nama unit, nama ketua, role, keterangan, email dan no telp/hp.	Super Admin
		AME_F_5903	Pada halaman detail akun juga terdapat seluruh list event yang pernah diajukan menggunakan akun tersebut.	Super Admin
		AME_F_5904	Aktor dapat mencari event, membatasi event yang ditampilkan dan menampilkan pagination yang diimplementasikan menggunakan *library* DataTables	Super Admin

Keterangan:

a. Approver: Dekan, WD1, WD2, WD3, KTU, BEM, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, TU Subbag Keuangan dan Kajur.

b. Event rutin merupakan event yang diselenggarakan harian/mingguan dalam rentang waktu tertentu. Bukan event rutin merupakan event yang diselenggarakan satu atau beberapa hari saja.

4.6 Verifikasi dan Validasi Kebutuhan

Penulis sebelumnya telah melakukan wawancara terhadap berbagai *stakeholder* yang tekait dengan pengajuan *event*, peminjaman *venue* dan pencairan dana di Fakultas Ilmu Administrasi Universitas Brawijaya Malang. Dari hasil wawancara tersebut kemudian penulis melakukan analisis yang menghasilkan gambaran umum aplikasi, identifikasi aktor, definisi kebutuhan fungsional dan definisi kebutuhan non-fungsional serta spesifikasi kebutuhan fungsional. Untuk memeriksa apakah hasil analisis tersebut benar, maka penulis melakukan tahap verifikasi dan validasi kebutuhan.

Pada penelitian ini, penulis melakukan validasi kebutuhan kepada kepala Gugus Jaminan Mutu Fakultas Ilmu Administrasi Universitas Brawijaya yaitu Bapak Dr. Drs. Muhammad Saifi, M.Si. Penulis melakukan validasi kebutuhan dengan cara membuat dokumen validasi kebutuhan secara terpisah kemudian penulis menemui Bapak Dr. Drs. Muhammad Saifi, M.Si secara langsung dan meminta beliau untuk memverifikasi *flowchart* yang telah dibuat berdasarkan hasil wawancara dengan beberapa stakeholder terkait pengajuan event, peminjaman venue dan pencairan dana di FIA UB. Hasil dari proses validasi dapat dilihat pada Lampiran J dan K.

Penulis juga melakukan verifikasi kebutuhan untuk memastikan bahwa kebutuhan telah dengan benar didefinisikan dan dispesifikasikan kebutuhan-kebutuhan yang ada di aplikasi Manajemen Event. Penulis melakukan verifikasi kebutuhan dengan cara memeriksa kebutuhan-kebutuhan tersebut menggunakan parameter-parameter verifikasi itu sendiri yaitu *readability*, *testability*, *completeness*, *identifiability*, dan *ambiguity*.

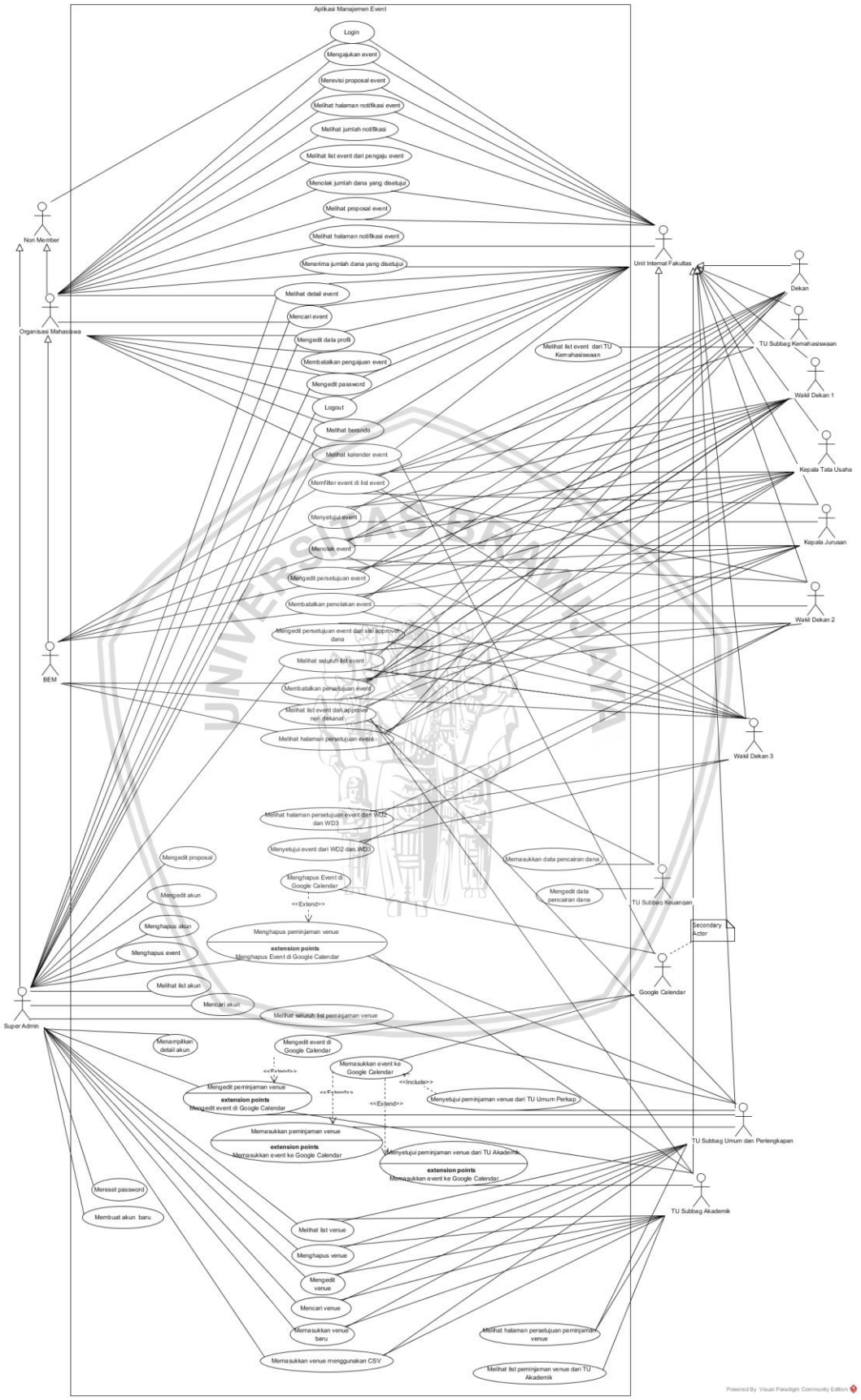
Setelah tahap verifikasi dan validasi selesai dilakukan maka penulis dapat memastikan bahwa hasil analisis yang telah dibuat penulis (identifikasi aktor, definisi kebutuhan fungsional dan definisi kebutuhan non-fungsional serta spesifikasi kebutuhan fungsional) adalah benar, *clear* serta *fixed* dan sesuai dengan prosedur pengajuan *event*, peminjaman *venue* dan pencairan dana yang berlaku di Fakultas Ilmu Administrasi Universitas Brawijaya Malang.

4.7 Pemodelan Kebutuhan

Pemodelan kebutuhan merupakan tahap yang dilakukan untuk memodelkan hasil analisis kebutuhan kedalam diagram. Pada penelitian ini, karena penulis menggunakan paradigma *object oriented* maka penulis akan menggunakan salah satu diagram yang termasuk UML diagram yaitu *use case diagram* untuk memodelkan kebutuhan pada penelitian ini. Untuk mendetailkan *use case diagram*, penulis akan menggunakan *use case scenario* yang akan dituliskan dalam format tabel. Kemudian penulis juga membuat State Transition Diagram yang menggambarkan alur serta status disposisi proposal atau persetujuan event pada penelitian ini.

4.7.1 Use Case Diagram

Pada diagram *use case* dibawah terdapat 16 aktor dan 59 *use cases*. Terdapat 15 *primary actor* dan 1 *secondary actor* yaitu Google Calendar. *Secondary Actor* merupakan aktor yang digunakan untuk membantu jalannya suatu *use case* sedangkan *Primary Actor* merupakan aktor yang dapat menjalankan *use case* tersebut. Google Calendar pada aplikasi ini digunakan untuk membantu jalannya *use case* melihat kalender, mengedit *event* di Google Calendar, menghapus *event* di Google Calendar dan memasukkan *event* ke Google Calendar. Selain itu, terdapat relasi *association*, *include* dan *extend* serta terdapat generalisasi. Gambar 4.9 dibawah ini merupakan ilustrasi dari *Use Case Diagram* pada penelitian ini.



Gambar 4.9 Diagram *Use Case*

4.7.2 Use Case Scenario

Use case scenario merupakan pendetailan dari setiap *use case* yang telah digambarkan sebelumnya. *Use case scenario* berisi aktor yang dapat mengakses *use case* tersebut, tujuan *use case*, prekondisi, *main flow*, *alternative flow* dan post kondisi.

a) *Use Case Scenario* Login

Login merupakan fungsionalitas yang disediakan sistem untuk mengautentikasi dan mengautorisasi aktor. *Use Case Scenario* login dapat dilihat pada tabel berikut.

Tabel 4.5 *Use Case Scenario* Login

Flow of Events for Login	
Kode Kebutuhan	AME_F_100
Objective	Untuk mengautentikasi dan mengautorisasi aktor
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan, Non Member
Pre-condition	1. Aktor belum masuk kedalam sistem 2. Aktor telah berada di halaman login
Main Flow	1. Aktor mengisi *field* NIP/NIM dan *password* serta menekan tombol login. 2. Sistem mengecek NIP/NIM dan *password* aktor di database. 3. Sistem mengarahkan aktor ke halaman beranda. 4. Sistem menampilkan notifikasi dengan pesan “Welcome, username”.
Alternative Flow	1. Jika *form login* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*. 2. Jika NIP/NIM tidak terdaftar maka aktor akan diarahkan ke halaman login dan sistem akan menampilkan pesan *error* “NIP/NIM yang anda masukkan salah”. 3. Jika *password* tidak sesuai dengan NIP/NIM yang dimasukkan maka aktor akan diarahkan ke halaman login dan sistem akan menampilkan pesan *error* “Password yang anda masukkan salah”.
Post-condition	Aktor telah berhasil diauthentikasi dan diauthorisasi.

b) *Use Case Scenario* Logout

Logout merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor jika ingin keluar dari sistem. *Use Case Scenario* logout dapat dilihat pada tabel berikut.

Tabel 4.6 *Use Case Scenario* Logout

Flow of Events for Logout	
Kode Kebutuhan	AME_F_200
Objective	Untuk mengeluarkan aktor dari sistem.
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diautentikasi dan diautorisasi
Main Flow	1. Aktor menekan tombol *logout*. 2. Sistem mengupdate *database session* dan mengarahkan aktor ke halaman login.
Alternative Flow	-
Post-condition	Aktor telah dideautentikasi dari sistem.

c) *Use Case Scenario* Melihat Beranda

Melihat beranda merupakan fungsionalitas yang digunakan untuk melihat halaman utama sistem (beranda). *Use Case Scenario* melihat beranda dapat dilihat pada tabel berikut.

Tabel 4.7 *Use Case Scenario* Melihat Beranda

Flow of Events for Melihat Beranda	
Kode Kebutuhan	AME_F_300
Objective	Untuk menampilkan halaman utama (beranda) sistem.
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diautentikasi dan diautorisasi
Main Flow	1. Aktor mengakses halaman beranda sistem. 2. Sistem menampilkan halaman beranda.
Alternative Flow	-

Post-condition	Aktor dapat melihat halaman utama (beranda).

d) *Use Case Scenario* Melihat List *Event dari Pengaju Event*

Melihat list *event* dari pengaju event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor khususnya yang memiliki *role* pengaju event untuk melihat list *event* yang pernah diajukan oleh aktor tersebut. *Use Case Scenario* melihat list *event* dari pengaju *event* dapat dilihat pada tabel berikut

Tabel 4.8 *Use Case Scenario* Melihat *List Event* dari Pengaju *Event*

Flow of Events for Melihat List *Event* dari Pengaju *Event*	
Kode Kebutuhan	AME_F_400
Objective	Untuk menampilkan list *event* yang pernah diajukan aktor
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas
Pre-condition	1. Aktor telah diautentikasi dan diautorisasi
Main Flow	1. Aktor menekan menu *list event* 2. Sistem membaca *database* dan menampilkan *list event*
Alternative Flow	1. Jika belum ada list event, maka sistem akan menampilkan pesan "*No data available in table*".
Post-condition	Aktor dapat melihat list event

e) *Use Case Scenario* Melihat Halaman Persetujuan Peminjaman *Venue*

Melihat Halaman Persetujuan Peminjaman *Venue* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat peminjaman venue yang memerlukan persetujuan dari aktor. *Use Case Scenario* Melihat Halaman Persetujuan Peminjaman Venue dapat dilihat pada tabel berikut.

Tabel 4.9 *Use Case Scenario* Melihat Halaman Persetujuan Peminjaman Venue

Flow of Events for Melihat Halaman Persetujuan Peminjaman Venue	
Kode Kebutuhan	AME_F_500
Objective	Untuk melihat peminjaman venue yang memerlukan persetujuan dari aktor
Aktor	TU Subbag Akademik, TU Subbag Umum dan Perlengkapan
Pre-condition	1. Aktor telah diautentikasi dan diauthorisasi 2. Aktor telah berada di tab persetujuan *event*
Main Flow	1. Sistem membaca data persetujuan *event* dari *database*

	2. Sistem menampilkan peminjaman *venue* yang memerlukan persetujuan
Alternative Flow	1. Jika tidak ada *event* yang memerlukan persetujuan maka sistem akan menampilkan notifikasi yang berisi pesan "Belum ada *event*"
Post-condition	Aktor dapat melihat *list* peminjaman *venue* yang memerlukan persetujuan dari aktor

f) *Use Case Scenario* Melihat Halaman Persetujuan *Event*

Melihat Halaman Persetujuan *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat list *event* yang memerlukan persetujuan aktor. *Use Case Scenario* Melihat Halaman Persetujuan *Event* dapat dilihat pada tabel berikut.

Tabel 4.10 *Use Case Scenario* Melihat Halaman Persetujuan *Event*

Flow of Events for Melihat Halaman Persetujuan *Event*	
Kode Kebutuhan	AME_F_600
Objective	Untuk menampilkan *list event* yang memerlukan persetujuan aktor
Aktor	WD1, KTU, BEM, Dekan, Kajur
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di tab persetujuan *event*
Main Flow	1. Sistem membaca data persetujuan *event* dari *database* 2. Sistem menampilkan *list event* yang memerlukan persetujuan
Alternative Flow	1. Jika tidak ada *event* yang memerlukan persetujuan maka sistem akan menampilkan notifikasi yang berisi pesan "Belum ada *event"*
Post-condition	Aktor dapat melihat list *event* yang memerlukan persetujuan aktor

g) *Use Case Scenario* Memfilter Event yang Ditampilkan di List Event

Memfilter Event yang Ditampilkan di List Event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk memfilter *event* yang terdapat di *list event*. *Use Case Scenario* Memfilter Event yang Ditampilkan di List Event dapat dilihat pada tabel berikut.

Tabel 4.11 *Use Case Scenario* Memfilter Event yang Ditampilkan di List Event

Flow of Events for Memfilter Event yang Ditampilkan di List Event	
Kode Kebutuhan	AME_F_700
Objective	Untuk memfilter event yang ditampilkan di list event
Aktor	TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi. 2. Aktor telah berada di halaman list event
Main Flow	1. Aktor memilih kategori "tampilkan hanya event saya" yang ada di *dropdown filter event* 2. Sistem menampilkan list *event* yang pernah diajukan aktor tersebut.
Alternative Flow	1. Jika belum ada *event* yang pernah diajukan aktor tersebut, maka sistem akan menampilkan pesan "*No data available in table*". 2. Jika aktor memilih kategori "Tampilkan seluruh *event*" dan aktor adalah Dekan, WD1, WD2 atau WD3 maka sistem akan menampilkan seluruh list event yang ada di sistem. 3. Jika aktor memilih kategori "Tampilkan seluruh event" dan aktor adalah Kajur, KTU, BEM, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, TU Subbag Keuangan atau TU Subbag Kemahasiswaan maka sistem akan menampilkan seluruh list *event* yang pernah diajukan oleh aktor tersebut dan event yang pernah mendapatkan persetujuan dari aktor tersebut..
Post-condition	Aktor telah berhasil memfilter *event* yang ditampilkan di *list event*

h) *Use Case Scenario* Melihat Halaman Persetujuan *Event* dari WD2 dan WD3

Melihat Halaman Persetujuan Event dari WD2 dan WD3 merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat *list event* yang memerlukan persetujuan aktor tersebut. *Use Case Scenario* Melihat Halaman Persetujuan *Event* dari WD2 dan WD3 dapat dilihat pada tabel berikut.

Tabel 4.12 *Use Case Scenario* Melihat Halaman Persetujuan *Event* dari WD2 dan WD3

Flow of Events for Melihat Halaman Persetujuan *Event* dari WD2 dan WD3	
Kode Kebutuhan	AME_F_800
Objective	Untuk menampilkan *list event* yang memerlukan persetujuan aktor
Aktor	WD2, WD3
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman pengajuan event
Main Flow	1. Aktor menekan tab persetujuan event 2. Sistem membaca data event dari database 3. Sistem menampilkan list event yang memerlukan persetujuan
Alternative Flow	1. Jika tidak ada event yang memerlukan persetujuan maka sistem akan menampilkan notifikasi yang berisi pesan “Belum ada event”.
Post-condition	Aktor dapat melihat list event yang memerlukan persetujuan aktor

i) *Use Case Scenario* Melihat Detail *Event*

Melihat detail event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor jika ingin melihat informasi detail dari suatu *event*. *Use Case Scenario* melihat detail *event* dapat dilihat pada tabel berikut.

Tabel 4.13 Use Case Scenario Melihat Detail *Event*

Flow of Events for Melihat Detail *Event*	
Kode Kebutuhan	AME_F_900
Objective	Untuk menampilkan informasi detail *event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah berada di halaman *list event, list* peminjaman *venue* atau persetujuan *event*
Main Flow	1. Aktor menekan tombol detail *event* 2. Sistem membaca data *event* dari *database* 3. Sistem menampilkan detail *event* yang dipilih oleh

	aktor
Alternative Flow	-
Post-condition	Aktor dapat melihat detail *event*.

j) *Use Case Scenario* Mencari *Event*

Mencari event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mencari *event* yang pernah diajukan oleh aktor tersebut atau pernah disetujui oleh aktor tersebut. *Use Case Scenario* mencari *event* dapat dilihat pada tabel berikut.

Tabel 4.14 *Use Case Scenario* Mencari *Event*

Flow of Events for Mencari Event	
Kode Kebutuhan	AME_F_1000
Objective	untuk mencari *event* dari *list event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi. 2. Aktor telah berada di halaman List Event
Main Flow	1. Aktor mengisi field pencarian *event*. 2. Sistem menampilkan *event* yang dicari oleh aktor.
Alternative Flow	1. Apabila *event* yang dicari aktor tidak ditemukan, maka sistem akan menampilkan pesan "*No matching records found*".
Post-condition	Aktor dapat menemukan event yang dicari

k) *Use Case Scenario* Mengajukan *Event*

Mengajukan *event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor jika ingin mengajukan *event* baru. *Use Case Scenario* mengajukan *event* dapat dilihat pada tabel berikut.

Tabel 4.15 Use Case Scenario Mengajukan *Event*

Flow of Events for Mengajukan *Event*	
Kode Kebutuhan	AME_F_1100
Objective	Untuk mengajukan *event* baru.
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag

	Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman pengajuan *event*
Main Flow	1. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, nama lengkap peminjam, NIP/NIM peminjam, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol ajukan *event* 2. Sistem melakukan validasi *form* pengajuan *event* disisi *client* 3. Sistem melakukan validasi *form* pengajuan *event* disisi *server* 4. Sistem mengecek ketersediaan *venue event* yang dipinjam 5. Sistem menyimpan data pengajuan *event,* pencairan dana dan peminjaman *venue* untuk event tidak rutin kedalam *database* 6. Aktor mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil mengajukan *event*".
Alternative Flow	1. Jika form pengajuan event tidak *valid* maka sistem akan menampilkan pesan berisi notifikasi *error*. 2. Jika aktor mengisi *field* di tab peminjaman venue untuk event tidak rutin, peminjaman venue untuk *event* rutin atau tidak membutuhkan *venue* secara bersamaan maka sistem akan menampilkan notifikasi berisi pesan *error* 3. Jika *venue* yang dimasukan telah direservasi oleh *event* lain, maka sistem akan mengarahkan aktor ke halaman pengajuan *event* dan sistem akan menampilkan notifikasi dengan pesan "Peminjaman venue bentrok dengan event dibawah ini" 4. Jika *event* merupakan *event* rutin maka sistem akan menyimpan data peminjaman *venue* rutin kedalam database 5. Jika *event* tidak membutuhkan *venue* maka sistem akan menyimpan data peminjaman *venue* sebagai "Diluar Kota Malang" atau "Didalam Kota Malang".

Post-condition	Aktor telah berhasil mengajukan *event*.

l) *Use Case Scenario* Mengedit Data Profil

Mengedit data profil merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah data akun miliknya seperti NIP/NIM, nama lengkap, email dan no. telepon. *Use Case Scenario* mengedit data profil dapat dilihat pada tabel berikut.

Tabel 4.16 *Use Case Scenario* Mengedit Data Profil

Flow of Events for Mengedit Data Profil	
Kode Kebutuhan	AME_F_1200
Objective	Untuk mengubah data profil aktor
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman edit profil
Main Flow	1. Aktor mengisi *field* NIP/NIM, nama unit/organisasi, nama ketua unit/organisasi, no. HP atau email dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit profil 3. Sistem menyimpan data edit profil kedalam *database* 4. Sistem mengarahkan aktor ke halaman edit profil dan menampilkan notifikasi berisi pesan “Berhasil mengedit data profil”
Alternative Flow	1. Jika aktor mengubah NIP/NIM dengan NIP/NIM yang telah ada di *database* maka sistem akan menampilkan notifikasi yang berisi pesan “NIP/NIM telah dipakai akun lain” 2. Jika aktor mengisi *field* dengan nilai yang tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*
Post-condition	Aktor telah berhasil mengubah data profil.

m) *Use Case Scenario* Mengedit *Password*

Mengedit *password* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor jika ingin mengubah *password* akun miliknya. *Use Case Scenario* mengedit password dapat dilihat pada tabel berikut.

Tabel 4.17 *Use Case Scenario* Mengedit Password

Flow of Events for Mengedit *Password*	
Kode Kebutuhan	AME_F_1300
Objective	Untuk mengubah *password* aktor
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi. 2. Aktor telah mengakses halaman edit profil dan memilih *tab* edit *password*.
Main Flow	1. Aktor mengisi *field password* lama, *password* baru dan konfirmasi *password* serta mengklik tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit *password* 3. Sistem menyimpan data edit *password* kedalam *database* 4. Sistem menampilkan aktor ke halaman edit profil dan menampilkan notifikasi berisi pesan "*Password* berhasil diubah".
Alternative Flow	1. Jika aktor mengisi *field* dengan nilai yang tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error* 2. Jika nilai field *password* lama tidak sesuai dengan *password* yang ada di *database* maka sistem akan menampilkan notifikasi berisi pesan *error* "Periksa kembali *password* yang anda masukkan" 3. Jika nilai pada *field password* baru tidak sama dengan nilai pada *field* konfirmasi *password* maka sistem akan menampilkan notifikasi berisi pesan *error* "*Password* baru dan konfirmasi *password* harus sama"
Post-condition	Aktor telah berhasil merubah *password*.

n) *Use Case Scenario* Melihat Jumlah Notifikasi

Melihat Jumlah Notifikasi merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat jumlah notifikasi. *Use Case Scenario* Melihat Jumlah Notifikasi dapat dilihat pada tabel berikut.

Tabel 4.18 *Use Case Scenario* Melihat Jumlah Notifikasi

Flow of Events for Melihat Jumlah Notifikasi	
Kode Kebutuhan	AME_F_1400
Objective	Untuk menampilkan jumlah notifikasi
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi
Main Flow	1. Aktor mengakses salah satu halaman yang ada di sistem 2. Sistem menampilkan jumlah notifikasi
Alternative Flow	-
Post-condition	Aktor dapat melihat jumlah notifikasi

o) *Use Case Scenario* Mengedit Peminjaman *Venue*

Mengedit Peminjaman *Venue* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi untuk mengedit *venue* yang dipinjam dan waktu peminjaman *venue*. *Use Case Scenario* Mengedit Peminjaman *Venue* dapat dilihat pada tabel berikut.

Tabel 4.19 *Use Case Scenario* Mengedit Peminjaman *Venue*

Flow of Events for Mengedit Peminjaman *Venue*	
Kode Kebutuhan	AME_F_1500
Objective	Untuk mengedit *venue* yang dipinjam dan waktu peminjaman *venue*
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi 2. Aktor telah berada di halaman edit peminjaman venue
Main Flow	1. Aktor mengisi *field* nama venue, tanggal peminjaman, waktu mulai atau waktu selesai dan

	menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit peminjaman *venue* 3. Sistem menyimpan data edit peminjaman *venue* kedalam *database* 4. Sistem menampilkan halaman edit peminjaman *venue* dan menampilkan notifikasi berisi pesan "Berhasil mengedit peminjaman *venue*"
Alternative Flow	1. Perluasan ke *use case* mengedit *event* di Google Calendar. 2. Jika *venue* yang ingin dipinjam akan digunakan oleh event lain maka sistem akan menampilkan notifikasi berisi pesan error "Peminjaman venue bentrok dengan event dibawah ini."
Post-condition	Aktor telah berhasil mengedit peminjaman *venue*

p) *Use Case Scenario* Membatalkan Persetujuan Event

Membatalkan Persetujuan *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk membatalkan persetujuan dari *event* yang sebelumnya telah ia setujui. *Use Case Scenario* Membatalkan Persetujuan *Event* dapat dilihat pada tabel berikut.

Tabel 4.20 *Use Case Scenario* Membatalkan Persetujuan *Event*

Flow of Events for Membatalkan Persetujuan Event	
Kode Kebutuhan	AME_F_1600
Objective	Untuk memfasilitasi aktor dalam membatalkan persetujuan dari *event* yang sebelumnya ia setujui
Aktor	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor mengakses halaman manajemen *list event*
Main Flow	1. Aktor mengklik *button* membatalkan persetujuan *event* 2. Sistem mengupdate data *event* 3. Sistem mengarahkan aktor ke halaman manajemen *list event* dan menampilkan notifikasi berisi pesan "Berhasil membatalkan persetujuan *event*"
Alternative Flow	-
Post-condition	Aktor telah berhasil membatalkan persetujuan *event*

q) *Use Case Scenario* Mengedit Persetujuan Event

Mengedit Persetujuan *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengedit persetujuan dari *event* yang sebelumnya ia setujui. *Use Case Scenario* Mengedit Persetujuan *Event* dapat dilihat pada tabel berikut.

Tabel 4.21 *Use Case Scenario* Mengedit Persetujuan Event

Flow of Events for Mengedit Persetujuan Event	
Kode Kebutuhan	AME_F_1700
Objective	Untuk mengedit data persetujuan dari *event* yang sebelumnya ia setujui
Aktor	WD1, KTU, BEM, Dekan, Kajur
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman edit persetujuan event
Main Flow	1. Aktor mengubah *field* catatan dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit persetujuan *event* 3. Sistem menyimpan perubahan data persetujuan *event* kedalam *database* 4. Sistem mengarahkan aktor ke halaman manajemen *list event* dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan *event*"
Alternative Flow	1. Jika aktor mengisi *field* dengan nilai yang tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*
Post-condition	Aktor telah berhasil mengedit data persetujuan *event*.

r) *Use Case Scenario* Mengedit Data Pencairan Dana

Mengedit Data Pencairan Dana merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengedit data pencairan dana dari *event* yang sebelumnya telah diinputkan aktor. *Use Case Scenario* Mengedit Data Pencairan Dana dapat dilihat pada tabel berikut.

Tabel 4.22 *Use Case Scenario* Mengedit Data Pencairan Dana

Flow of Events for Mengedit Data Pencairan Dana	
Kode Kebutuhan	AME_F_1800
Objective	Untuk mengedit data pencairan dana dari suatu *event*

Aktor	TU Subbag Keuangan
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman edit pencairan dana
Main Flow	1. Aktor mengubah *field* nama pengambil dana, NIP/NIM pengambil dana, catatan atau tanggal pengambilan dana dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit pencairan dana 3. Sistem menyimpan data pencairan dana kedalam *database* 4. Sistem mengarahkan aktor ke halaman manajemen *list event* dan menampilkan notifikasi berisi pesan "Berhasil mengedit *event*"
Alternative Flow	1. Jika *form* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*
Post-condition	Aktor telah berhasil mengedit data pencairan dana

s) *Use Case Scenario* Mengedit Persetujuan *Event* dari Approver Dana

Mengedit Persetujuan *Event* dari Approver Dana merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor khususnya aktor yang menyetujui jumlah dana yang dicairkan untuk mengedit data persetujuan dari *event* yang sebelumnya ia setujui. *Use Case Scenario* Mengedit Persetujuan *Event* dari Approver Dana dapat dilihat pada tabel berikut.

Tabel 4.23 *Use Case Scenario* Mengedit Persetujuan *Event* dari Approver Dana

Flow of Events for Mengedit Persetujuan Event dari Approver Dana	
Kode Kebutuhan	AME_F_1900
Objective	Untuk mengedit data persetujuan dari *event* yang sebelumnya disetujui oleh aktor
Aktor	WD2, WD3
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman edit persetujuan event
Main Flow	1. Aktor mengubah *field* jumlah dana yang disetujui atau catatan dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit persetujuan *event* 3. Sistem menyimpan perubahan kedalam *database* 4. Sistem mengarahkan aktor ke halaman

	manajemen *list event* dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan *event*"
Alternative Flow	1. Jika aktor mengisi *field* dengan nilai yang tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*
Post-condition	Aktor telah berhasil mengedit data persetujuan *event*

t) *Use Case Scenario* Melihat Proposal Event

Melihat Proposal Event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat proposal dari suatu *event*. *Use Case Scenario* Melihat Proposal *Event* dapat dilihat pada tabel berikut.

Tabel 4.24 *Use Case Scenario* Melihat Proposal Event

Flow of Events for Melihat Proposal Event	
Kode Kebutuhan	AME_F_2000
Objective	Untuk menampilkan *file* proposal dari suatu event
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi 2. Aktor telah berada di halaman detail *event*, *list event*, persetujuan atau peminjaman *venue*.
Main Flow	1. Aktor menekan *link* file proposal 2. Sistem mengarahkan aktor ke *tab browser* yang baru dan menampilkan file proposal pada *tab* tersebut
Alternative Flow	-
Post-condition	Aktor dapat melihat proposal *event*

u) *Use Case Scenario* Melihat Halaman Notifikasi *Event*

Melihat Halaman Notifikasi *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat notifikasi dari suatu event. Notifikasi tersebut dapat berisi *field* untuk mengedit file proposal (jika *event* ditolak) atau field untuk menyetujui atau menolak dana yang disetujui oleh approver dana (jika dana yang disetujui oleh approver dana kurang dari dana yang diajukan oleh pengaju *event*). *Use Case Scenario* Melihat Halaman Notifikasi *Event* dapat dilihat pada tabel berikut.

Tabel 4.25 *Use Case Scenario* Melihat Halaman Notifikasi *Event*

Flow of Events for Melihat Halaman Notifikasi *Event*	
Kode Kebutuhan	AME_F_2100
Objective	Untuk menampilkan halaman notifikasi *event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi 2. Aktor telah berada di halaman pengajuan event
Main Flow	1. Aktor mengklik *tab* notifikasi *event* 2. Sistem membaca data dari *database* 3. Sistem menampilkan halaman notifikasi *event*
Alternative Flow	1. Jika tidak ada notifikasi *event*, maka sistem akan menampilkan notifikasi berisi pesan "Belum ada notifikasi".
Post-condition	Aktor dapat melihat notifikasi *event*

v) *Use Case Scenario* Menerima Jumlah Dana yang Disetujui

Menerima Jumlah Dana yang Disetujui merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menyetujui jumlah dana yang sebelumnya telah diinputkan *approver* dana. *Use Case Scenario* Menerima Jumlah Dana yang Disetujui dapat dilihat pada tabel berikut.

Tabel 4.26 *Use Case Scenario* Menerima Jumlah Dana yang Disetujui

Flow of Events for Menerima Jumlah Dana yang Disetujui	
Kode Kebutuhan	AME_F_2200
Objective	Untuk menyetujui jumlah dana yang sebelumnya telah disetujui oleh *approver* dana
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diautentikasi dan diauthorisasi 2. Aktor telah berada di halaman notifikasi event
Main Flow	1. Aktor mengklik button setujui 2. Sistem menyimpan data penerimaan dana kedalam database 3. Sistem mengarahkan aktor ke halaman pengajuan

	event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui jumlah dana"
Alternative Flow	-
Post-condition	Untuk menyetujui jumlah dana yang sebelumnya telah disetujui oleh *approver* dana

w) *Use Case Scenario* Menolak Jumlah Dana yang Disetujui

Menolak Jumlah Dana yang Disetujui merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menolak jumlah dana yang sebelumnya telah diinputkan *approver* dana. *Use Case Scenario* Menolak Jumlah Dana yang Disetujui dapat dilihat pada tabel berikut.

Tabel 4.27 *Use Case Scenario* Menolak Jumlah Dana yang Disetujui

Flow of Events for Menolak Jumlah Dana yang Disetujui	
Kode Kebutuhan	AME_F_2300
Objective	Untuk menolak jumlah dana yang disetujui oleh *approver* dana
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman notifikasi event
Main Flow	1. Aktor mengklik *button* tolak 2. Sistem menyimpan data penolakan dana 3. Sistem mengarahkan aktor ke halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menolak jumlah dana".
Alternative Flow	-
Post-condition	Aktor telah berhasil menolak dana yang disetujui oleh *approver* dana

x) *Use Case Scenario* Merevisi Proposal Event

Merevisi Proposal Event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk merevisi proposal *event* yang sebelumnya ditolak oleh *approver*. *Use Case Scenario* Merevisi Proposal *Event* dapat dilihat pada tabel berikut.

Tabel 4.28 *Use Case Scenario* Merevisi Proposal *Event*

Flow of Events for Merevisi Proposal *Event*	
Kode Kebutuhan	AME_F_2400
Objective	Untuk merevisi proposal *event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diauthentikasi dan diauthorisasi 2. Aktor telah berada di halaman notifikasi event
Main Flow	1. Aktor memilih file proposal dan mengklik tombol simpan perubahan 2. Sistem menyimpan file proposal dan mengupdate database serta mengarahkan aktor ke halaman pengajuan event kemudian menampilkan notifikasi berisi pesan "Berhasil merevisi proposal"
Alternative Flow	1. Jika aktor tidak mengisi field namun menekan tombol simpan perubahan maka sistem akan menampilkan notifikasi berisi pesan error "Pilih file"
Post-condition	Aktor telah berhasil merevisi proposal *event*

y) *Use Case Scenario* Melihat Kalender *Event*

Melihat kalender *event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat kalender *event*. Kalender yang ditampilkan tersebut berasal dari aplikasi Google Calendar. *Use Case Scenario* melihat kalender *event* dapat dilihat pada tabel berikut.

Tabel 4.29 *Use Case Scenario* Melihat Kalender *Event*

Flow of Events for Melihat Kalender *Event*	
Kode Kebutuhan	AME_F_2500
Objective	Untuk menampilkan kalender *event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Super Admin, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan, Google Calendar (*Secondary actor*)
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi

	2. Aktor telah berada di halaman pengajuan *event*
Main Flow	1. Aktor menekan *tab* kalender *event* 2. Sistem menampilkan kalender *event*
Alternative Flow	-
Post-condition	Aktor dapat melihat kalender *event*

z) *Use Case Scenario* Melihat List Event dari TU Kemahasiswaan

Melihat List *Event* dari TU Kemahasiswaan merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat seluruh *list event* yang pernah diajukan oleh aktor tersebut dan *list event* yang pernah diajukan oleh organisasi mahasiswa. *Use Case Scenario* Melihat *List Event* dari TU Kemahasiswaan dapat dilihat pada tabel berikut.

Tabel 4.30 *Use Case Scenario* Melihat List Event dari TU Kemahasiswaan

Flow of Events for Melihat List Event dari TU Kemahasiswaan	
Kode Kebutuhan	AME_F_2600
Objective	Untuk menampilkan *list event* yang pernah diajukan aktor dan *list event* yang pernah diajukan oleh organisasi mahasiswa
Aktor	TU Subbag Kemahasiswaan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi
Main Flow	1. Aktor mengakses halaman *list event* 2. Sistem membaca data *event* dari database 3. Sistem menampilkan *list event*
Alternative Flow	1. Jika belum ada list *event*, maka sistem akan menampilkan pesan "*No data available in table*"
Post-condition	Aktor dapat melihat *list event*

aa) *Use Case Scenario* Memasukkan *Event* ke Google Calendar

Memasukkan *event* ke Google Calendar merupakan perluasan fungsi dari fungsionalitas Menyetujui Peminjaman *Venue* dari TU Akademik dan Memasukkan Peminjaman *Venue*. Fungsionalitas ini digunakan untuk mengirimkan data *event* kedalam Google Calendar. *Use Case Scenario* memasukkan *event* ke Google Calendar dapat dilihat pada tabel berikut.

Tabel 4.31 *Use Case Scenario* Memasukkan Event ke Google Calendar

Flow of Events for Memasukkan Event ke Google Calendar	
Kode Kebutuhan	AME_F_2700

Objective	Untuk memasukkan data peminjaman *venue* ke Google Calendar
Aktor	TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, Google Calendar (*Secondary Actor*)
Pre-condition	1. Aktor telah diautentikasi dan diauthorisasi 2. Aktor telah menyetujui peminjaman *venue* atau data peminjaman *venue* berhasil disimpan
Main Flow	1. Sietem mengirimkan data peminjaman *venue* ke Google Calender 2. Sistem mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menyetujui *event*".
Alternative Flow	-
Post-condition	Data peminjaman *venue* berhasil dimasukkan ke Google Calendar.

bb) *Use Case Scenario* Melihat List *Event* dari Approver Non Dekanat

Melihat *List Event* dari Approver Non Dekanat merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat *list event* yang pernah diajukan oleh aktor tersebut dan *event* yang perlu mendapatkan persetujuan dari aktor tersebut. *Use Case Scenario* Melihat List *Event* dari Approver Non Dekanat dapat dilihat pada tabel berikut.

Tabel 4.32 *Use Case Scenario* Melihat List *Event* dari Approver Non Dekanat

Flow of Events for Melihat List *Event* dari Approver Non Dekanat	
Kode Kebutuhan	AME_F_2800
Objective	Untuk menampilkan list *event* dari sisi Approver Non Dekanat
Aktor	Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah diautentikasi dan diautorisasi
Main Flow	1. Aktor menekan menu list *event* 2. Sistem membaca data *event* dari *database* 3. Sistem menampilkan *list event*
Alternative Flow	1. Jika belum ada list event, maka sistem akan menampilkan pesan "*No data available in table*".
Post-condition	Aktor dapat melihat *list event*

cc) *Use Case Scenario* Mengedit Proposal

Mengedit proposal merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah proposal event. *Use Case Scenario* mengedit proposal dapat dilihat pada tabel berikut.

Tabel 4.33 Use Case Scenario Mengedit Proposal

Flow of Events for Mengedit Proposal	
Kode Kebutuhan	AME_F_2900
Objective	Untuk mengubah proposal event
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman edit event dan memilih tab edit proposal
Main Flow	1. Aktor memilih file proposal dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit proposal 3. Sistem menyimpan file proposal dan meng*update database* 4. Sistem menampilkan halaman edit *event* dan menampilkan notifikasi berisi pesan "Berhasil mengedit proposal *event*"
Alternative Flow	1. Jika aktor tidak mengisi *field* file proposal namun menekan tombol simpan perubahan maka sistem akan menampilkan notifikasi berisi pesan *error*
Post-condition	Aktor berhasil mengedit proposal *event*

dd) *Use Case Scenario* Membatalkan Penolakan Event

Membatalkan Penolakan Event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk membatalkan event yang sebelumnya ditolak oleh aktor terkait. *Use Case Scenario* Membatalkan Penolakan Event dapat dilihat pada tabel berikut.

Tabel 4.34 *Use Case Scenario* Membatalkan Penolakan Event

Flow of Events for Membatalkan Penolakan Event	
Kode Kebutuhan	AME_F_3000
Objective	Untuk membatalkan *event* yang sebelumnya ditolak oleh aktor terkait
Aktor	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM

Pre-condition	1. Aktor telah diautentikasi dan diautorisasi 2. Aktor telah mengakses halaman *List Event*
Main Flow	1. Aktor menekan tombol batalkan penolakan *event* 2. Sistem mengarahkan aktor ke halaman list *event* dan menampilkan notifikasi yang berisi pesan "Berhasil membatalkan penolakan *event*"
Alternative Flow	-
Post-condition	Aktor telah berhasil membatalkan penolakan *event*

ee) *Use Case Scenario* Menyetujui *Event*

Menyetujui *event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menyetujui proposal *event* yang diajukan. *Use Case Scenario* menyetujui event dapat dilihat pada tabel berikut.

Tabel 4.35 *Use Case Scenario* Menyetujui *Event*

Flow of Events for Menyetujui *Event*	
Kode Kebutuhan	AME_F_3100
Objective	Untuk menyetujui *event*
Aktor	Dekan, WD1, KTU, BEM, Kajur
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi 2. Aktor telah berada di halaman persetujuan *event*
Main Flow	1. Aktor mengisi *field* catatan dan menekan tombol persetujuan *event* 2. Sistem melakukan validasi *form* persetujuan *event* 3. Sistem menyimpan data persetujuan *event* ke *database*. 4. Sistem mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menyetujui *event*"
Alternative Flow	1. Jika *form* persetujuan *event* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor telah berhasil menyetujui *event*

ff) *Use Case Scenario* Membatalkan Pengajuan *Event*

Membatalkan Pengajuan Event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk membatalkan pengajuan *event* yang sebelumnye telah ditolak oleh aktor yang bertindak sebagai *approver*. *Use Case Scenario* Membatalkan Pengajuan Event dapat dilihat pada tabel berikut.

Tabel 4.36 *Use Case Scenario* Membatalkan Pengajuan *Event*

Flow of Events for Membatalkan Pengajuan *Event*	
Kode Kebutuhan	AME_F_3200
Objective	Untuk membatalkan pengajuan *event*
Aktor	Organisasi Mahasiswa, Unit Internal Fakultas, TU Subbag Kemahasiswaan, Dekan, WD1, WD2, WD3, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman notifikasi event
Main Flow	1. Aktor mengklik tombol batalkan pengajuan event 2. Sistem menghapus data event dari database 3. Sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil membatalkan pengajuan event"
Alternative Flow	-
Post-condition	Aktor telah berhasil membatalkan pengajuan *event*

gg) *Use Case Scenario* Menyetujui *Event* dari WD2 dan WD3

Menyetujui event dari WD2 dan WD3 merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menyetujui proposal event dan menentukan jumlah dana yang dicairkan jika event yang diajukan tersebut membutuhkan dana dari fakultas/kemahasiswaan serta mengizinkan pengajuan peminjaman venue. *Use Case Scenario* menyetujui event dari WD2 dan WD3 dapat dilihat pada tabel berikut.

Tabel 4.37 *Use Case Scenario* Menyetujui *Event* dari WD2 dan WD3

Flow of Events for Menyetujui *Event* dari WD2 dan WD3	
Kode Kebutuhan	AME_F_3300
Objective	Untuk menyetujui *event* dari WD2 dan WD3
Aktor	WD2, WD3
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman persetujuan event
Main Flow	1. Aktor mengisi *checkbox* perizinan peminjaman venue, field jumlah dana yang disetujui dan field catatan dan menekan tombol persetujuan *event* 2. Sistem melakukan validasi terhadap *form* persetujuan event

	3. Sistem menyimpan data persetujuan kedalam database 4. Sistem mengarahkan aktor ke halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui *event*"
Alternative Flow	1. Jika *form* persetujuan *event* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*. 2. Jika *event* tidak membutuhkan dana atau dana tidak berasal dari fakultas/kemahasiswaan maka aktor tidak mengisi field jumlah dana yang disetujui. 3. Jika *event* tidak membutuhkan *venue* maka aktor tidak mengisi field *checkbox* perizinan peminjaman *venue*.
Post-condition	Aktor telah berhasil menyetujui event

hh) *Use Case Scenario* Menolak *Event*

Menolak *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi Approver untuk menolak suatu *event*. *Use Case Scenario* menolak *event* dapat dilihat pada tabel berikut.

Tabel 4.38 *Use Case Scenario* Menolak *Event*

Flow of Events for Menolak *Event*	
Kode Kebutuhan	AME_F_3400
Objective	Untuk menolak event
Aktor	Dekan, WD1, WD2, WD3, Kajur, KTU, BEM
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman persetujuan event
Main Flow	1. Aktor mengisi *field* catatan dan menekan tombol penolakan *event* 2. Sistem memvalidasi *form* penolakan *event* 3. Sistem menampilkan *alert* konfirmasi 4. Aktor menekan tombol "Oke". 5. Sistem menyimpan data penolakan *event* kedalam *database* 6. Sistem mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menolak *event*"
Alternative Flow	1. Jika *form* persetujuan *event* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan

	error.
Post-condition	Aktor telah berhasil menolak *event*

ii) *Use Case Scenario* Memasukkan Data Pencairan Dana

Memasukkan data pencairan dana merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk memasukkan data saat proses pencairan dana seperti nama pengambil dana dan NIM/jabatan pengambil dana. *Use Case Scenario* memasukkan data pencairan dana dapat dilihat pada tabel berikut.

Tabel 4.39 *Use Case Scenario* Memasukkan Data Pencairan Dana

Flow of Events for Memasukkan Data Pencairan Dana	
Kode Kebutuhan	AME_F_3500
Objective	Untuk memasukkan data pencairan dana
Aktor	TU Subbag Keuangan
Pre-condition	1. Aktor telah terautheticasi dan terauthorisasi. 2. Aktor telah berada tab persetujuan event.
Main Flow	1. Aktor mengisi *checkbox* SPJ/LPJ, Nama Pengambil Dana, NIM/NIP pengambil dana dan catatan serta menekan tombol simpan 2. Sistem melakukan validasi terhadap *form* pencairan dana *event* 3. Sistem menyimpan data pencairan dana ke dalam *database* 4. Sistem mengarahkan aktor ke halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data pencairan dana"
Alternative Flow	1. Jika *form* data pencairan dana tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor berhasil menyimpan data pencairan dana.

jj) *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Umum Perkap

Menyetujui Peminjaman *Venue* dari TU Umum Perkap merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menyetujui peminjaman venue. *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Umum Perkap dapat dilihat pada tabel berikut.

Tabel 4.40 *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Umum Perkap

Flow of Events for Menyetujui Peminjaman *Venue* dari TU Umum Perkap	
Kode Kebutuhan	AME_F_3600
Objective	Untuk menyetujui peminjaman *venue* penyelenggaraan *event*
Aktor	TU Subbag Umum dan Perlengkapan
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah berada di halaman persetujuan *event*
Main Flow	1. Aktor mengisi *field* catatan dan menekan tombol setujui 2. Sistem melakukan validasi *form* persetujuan peminjaman *venue* 3. Sistem menyimpan data persetujuan peminjaman *venue* untuk *event* tidak rutin kedalam *database* dan mengirimkan data peminjaman *venue event* tidak rutin ke Google Calendar 4. Sistem mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan berhasil "Berhasil menyetujui peminjaman *venue*"
Alternative Flow	1. Jika *form* persetujuan tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error* 2. Jika peminjaman *venue* adalah peminjaman *venue* untuk *event* rutin maka sistem akan menyimpan data persetujuan peminjaman *venue* untuk *event* rutin kedalam *database* dan mengirimkan data peminjaman *venue event* rutin ke Google Calendar
Post-condition	Aktor telah berhasil menyetujui peminjaman *venue*

kk) *Use Case Scenario* Memasukkan Peminjaman *Venue*

Memasukkan peminjaman *venue* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk memasukkan data peminjaman *venue* secara langsung. Fungsionalitas ini dapat digunakan jika peminjam *venue* merupakan organisasi/lembaga diluar FIA UB atau jika peminjaman *venue* dapat dilakukan tanpa persetujuan Wakil Dekan 3 atau Wakil Dekan 2. *Use Case Scenario* memasukkan peminjaman *venue* dapat dilihat pada tabel berikut.

Tabel 4.41 *Use Case Scenario* Memasukkan Peminjaman *Venue*

Flow of Events for Memasukkan Peminjaman *Venue*	
Kode Kebutuhan	AME_F_3700

Objective	Untuk memasukkan data peminjaman *venue*
Aktor	TU Subbag Umum dan Perlengkapan
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Peminjaman Venue
Main Flow	1. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, nama lengkap peminjam, NIP/NIM peminjam, email, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol ajukan *event*. 2. Sistem melakukan validasi terhadap *form* peminjaman *venue*. 3. Sistem memeriksa ketersediaan *venue* yang dipinjam 4. Sistem menyimpan data peminjaman *venue* ke database 5. Sistem menampilkan halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Alternative Flow	1. Jika form peminjaman *venue* tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*. 2. Perluasan ke *use case* memasukkan *event* ke Google Calendar. 3. Jika aktor ingin meminjam *venue* dengan tanggal peminjaman atau waktu peminjaman yang berbeda maka aktor menekan tombol tambah peminjaman dan mengisikan *field venue* yang dipinjam, tanggal peminjaman, waktu mulai dan waktu selesai. 4. Jika pemijaman venue bentrok dengan peminjaman venue lain maka sistem akan menampikan notifikasi berisi pesan "Peminjaman venue bentrok dengan: Data peminjaman venue yang bentrok"
Post-condition	Aktor telah berhasil memasukkan data peminjaman *venue*

II) *Use Case Scenario* Melihat Seluruh List *Event*

Melihat Seluruh List *Event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat seluruh daftar *venue* yang ada di sistem. *Use Case Scenario* Melihat Seluruh *List Event* dapat dilihat pada tabel berikut.

Tabel 4.42 *Use Case Scenario* Melihat Seluruh *List Event*

Flow of Events for Melihat Seluruh *List Event*	
Kode Kebutuhan	AME_F_3800
Objective	Untuk menampilkan seluruh *list event*
Aktor	Super Admin, Dekan, WD1, WD2, WD3
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi
Main Flow	1. Aktor mengakses halaman *List Event* 2. Sistem membaca data *event* dari *database* 3. Sistem menampilkan *list event*
Alternative Flow	1. Jika tidak ada *venue* yang ditemukan maka sistem akan menampilkan pesan "*No data available in table*".
Post-condition	Aktor dapat melihat seluruh *list event*

mm) *Use Case Scenario* Menghapus *Event* di Google Calendar

Menghapus *event* di Google Calendar merupakan fungsionalitas yang disediakan sistem untuk menghapus *event* dari Google Calendar. *Use Case Scenario* menghapus *event* di Google Calendar dapat dilihat pada tabel berikut.

Tabel 4.43 Use Case Scenario Menghapus *Event* di Google Calendar

Flow of Events for Menghapus *Event* di Google Calendar	
Kode Kebutuhan	AME_F_3900
Objective	Untuk menghapus event di google calendar
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, Google Calendar (*Secondary Actor*)
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Penghapusan peminjaman *venue* di *database* telah berhasil
Main Flow	1. Sistem mengirimkan data hapus ke Google Calendar. 2. Sistem menampilkan notifikasi berisi pesan "Berhasil menghapus *event*"
Alternative Flow	-
Post-condition	Aktor telah berhasil menghapus *event* dari Google Calendar.

nn) *Use Case Scenario* Melihat List Venue

Melihat *List Venue* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat seluruh *list venue* yang ada di sistem. *Use Case Scenario* melihat *list venue* dapat dilihat pada tabel berikut.

Tabel 4.44 *Use Case Scenario* Melihat *List Venue*

Flow of Events for Melihat *List Venue*	
Kode Kebutuhan	AME_F_4000
Objective	Untuk menampilkan *list venue*
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi
Main Flow	1. Aktor mengakses halaman Manajemen *List Venue* 2. Sistem membaca data *venue* dari *database* 3. Sistem menampilkan seluruh list *venue*
Alternative Flow	1. Jika tidak ada venue yang ditemukan maka sistem akan menampilkan pesan "*No data available in table*".
Post-condition	Sistem akan menampilkan seluruh *list venue*

oo) *Use Case Scenario* Memasukkan Venue Baru

Memasukkan venue baru merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk memasukkan venue baru kedalam daftar venue. *Use Case Scenario* memasukkan venue baru dapat dilihat pada tabel berikut.

Tabel 4.45 *Use Case Scenario* Memasukkan Venue Baru

Flow of Events for Memasukkan Venue Baru	
Kode Kebutuhan	AME_F_4100
Objective	Untuk memasukkan data venue yang baru
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen List *Venue* dan menekan tombol tambah *venue*
Main Flow	1. Aktor mengisi *field* nama venue dan keterangan serta menekan tombol simpan 2. Sistem melakukan validasi terhadap *form* penambahan *venue* 3. Sistem menyimpan data *venue* kedalam *database*

	4. Sistem menampilkan halaman manajemen *list venue* dan menampilkan notifikasi berisi pesan "Berhasil memasukkan *venue*"
Alternative Flow	1. Jika nama *venue* yang dimasukkan telah ada didalam database maka sistem akan menampilkan notifikasi yang berisi pesan "Nama *venue* telah ada di dalam *database*" 2. Jika *form* memasukkan *venue* baru tidak *valid* maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor telah berhasil memasukkan *venue* baru

pp) *Use Case Scenario* Memasukkan Venue Menggunakan CSV

Memasukkan Venue Menggunakan CSV merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk memasukkan *venue* baru kedalam *list venue* menggunakan *file* CSV. *Use Case Scenario* Memasukkan *Venue* Menggunakan CSV dapat dilihat pada tabel berikut.

Tabel 4.46 *Use Case Scenario* Memasukkan Venue Menggunakan CSV

Flow of Events for Memasukkan Venue Menggunakan CSV	
Kode Kebutuhan	AME_F_4200
Objective	Untuk memasukkan data *venue* yang baru menggunakan file CSV
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen List Venue dan menekan tombol import venue menggunakan CSV
Main Flow	1. Aktor memilih file CSV dan menekan tombol simpan 2. Sistem memvalidasi *form upload* CSV 3. Sistem meng*upload file* CSV 4. Sistem membaca file CSV dan menyimpan data kedalam *database* 5. Sistem menampilkan halaman manajemen list *venue* dan menampilkan notifikasi berisi pesan "Berhasil meng*import venue* menggunakan CSV"
Alternative Flow	1. Jika *format* CSV tidak sesuai, maka sistem akan menampilkan notifikasi berisi pesan *error* "Perikasa kembali format file CSV yang anda

	upload". 2. Jika *form import* menggunanakn CSV dikosongkan namun aktor menekan tombol simpan maka sistem menampilkan notifikasi berisi pesan *error* "Pilih file".
Post-condition	Aktor telah berhasil memasukkan *venue* baru menggunakan file CSV

qq) *Use Case Scenario* Menghapus *Venue*

Menghapus venue merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menghapus *venue*. *Use Case Scenario* menyetujui menghapus *venue* dapat dilihat pada tabel berikut.

Tabel 4.47 *Use Case Scenario* Menghapus Venue

Flow of Events for Menghapus *Venue*	
Kode Kebutuhan	AME_F_4300
Objective	Untuk menghapus *venue*
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen *List Venue*
Main Flow	1. Aktor menekan tombol hapus 2. Sistem menghapus data *venue* dari database 3. Sistem menampilkan halaman manajemen *list venue* dan menampilkan notifikasi berisi pesan "Berhasil menghapus *venue*"
Alternative Flow	1. Jika aktor menekan tombol batal pada *alert* konfirmasi maka proses penghapusan akan dibatalkan.
Post-condition	Aktor berhasil menghapus *venue* dari *database*

rr) *Use Case Scenario* Mengedit *Venue*

Mengedit *venue* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah data *venue* seperti nama *venue* dan keterangan *venue*. *Use Case Scenario* mengedit *venue* dapat dilihat pada tabel berikut.

Tabel 4.48 *Use Case Scenario* Mengedit *Venue*

Flow of Events for Mengedit *Venue*

Kode Kebutuhan	AME_F_4400
Objective	Untuk mengubah data *venue*
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah berada di halaman edit *venue*
Main Flow	1. Aktor mengubah *field* nama *venue* atau keterangan dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit *venue* 3. Sistem mengedit data *venue* di database 4. Sistem menampilkan halaman manajemen list *venue* dan menampilkan notifikasi berisi pesan "Berhasil mengedit *venue*"
Alternative Flow	1. Jika aktor mengganti nama *venue* dan nama *venue* yang dimasukkan telah ada didalam *database* maka sistem akan menampilkan notifikasi yang berisi pesan "Nama *venue* telah ada di dalam database" 2. Jika form edit venue tidak valid maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor berhasil mengedit data *venue*

ss) *Use Case Scenario* Mencari Venue

Mencari venue merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mencari venue dari daftar venue. *Use Case Scenario* mencari venue dapat dilihat pada tabel berikut.

Tabel 4.49 *Use Case Scenario* Mencari Venue

Flow of Events for Mencari Venue	
Kode Kebutuhan	AME_F_4500
Objective	Untuk mencari venue dari daftar venue.
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen *List Venue*.
Main Flow	1. Aktor mengisi *field* pencarian *venue*

	2. Sistem menampilkan venue yang dicari aktor
Alternative Flow	1. Jika venue yang dicari tidak ditemukan, maka sistem akan menampilkan pesan "*No matching records found*".
Post-condition	Aktor dapat melihat *event* yang dicari

tt) *Use Case Scenario* Mengedit *Event* di Google Calendar

Mengedit event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah data *event* di Google Calendar. *Use Case Scenario* mengedit *event* dapat dilihat pada tabel berikut.

Tabel 4.50 Use Case Scenario Mengedit *Event* di Google Calendar

Flow of Events for Mengedit *Event* di Google Calendar	
Kode Kebutuhan	AME_F_4600
Objective	Untuk mengedit data *event* di Google Calendar
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, Google Calendar (*Secondary Actor*)
Pre-condition	1. Aktor telah terautheticasi dan terauthorisasi 2. Pengeditan data peminjaman venue di database telah dilakukan.
Main Flow	1. Sistem mengirimkan data edit ke Google Calender. 2. Sistem menampilkan notifikasi berisi pesan "Berhasil mengedit *event*"
Alternative Flow	-
Post-condition	Aktor telah berhasil menyimpan perubahan data *event* di Google Calendar.

uu) *Use Case Scenario* Melihat List Peminjaman Venue dari TU Akademik

Melihat List Peminjaman Venue dari TU Akademik merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi TU Akademik untuk melihat list peminjaman venue. *Use Case Scenario* Melihat List Peminjaman Venue dari TU Akademik dapat dilihat pada tabel berikut.

Tabel 4.51 *Use Case Scenario* Melihat List Peminjaman Venue dari TU Akademik

Flow of Events for Melihat List Peminjaman Venue dari TU Akademik	
Kode Kebutuhan	AME_F_4700
Objective	Untuk menampilkan list peminjaman *venue*

Aktor	TU Akademik
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi
Main Flow	1. Aktor mengakses halaman list peminjaman *venue* 2. Sistem membaca data peminjaman *venue* dari *database* 3. Sistem menampilkan *list* peminjaman *venue*
Alternative Flow	1. Jika belum ada daftar *event*, maka sistem akan menampilkan pesan "*No data available in table*"
Post-condition	Aktor dapat melihat *list* peminjaman *venue*

vv) *Use Case Scenario* Membuat Akun

Membuat akun baru merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk membuat akun baru atau sebagai mekanisme register pada aplikasi ini. *Use Case Scenario* membuat akun baru dapat dilihat pada tabel berikut.

Tabel 4.52 *Use Case Scenario* Membuat Akun Baru

Flow of Events for Membuat Akun Baru	
Kode Kebutuhan	AME_F_4800
Objective	Untuk membuat akun baru
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen *List* Akun dan menekan tombol tambah akun
Main Flow	1. Aktor mengisi *field* NIM/NIP, nama unit, nama ketua unit, email, no. hp, *role*, keterangan dan menekan tombol *simpan* 2. Sistem melakukan validasi terhadap *form* tambah akun 3. Sistem menyimpan data akun ke *database* 4. Sistem mengirimkan email ke email pembuat akun kemudian menampilkan halaman manajemen list akun dan notifikasi berisi pesan "Berhasil menambahkan akun"
Alternative Flow	1. Jika form tambah akun tidak valid maka sistem akan menampilkan notifikasi berisi pesan error. 2. Jika NIP/NIM yang dimasukkan telah dipakai aktor lain maka sistem akan menampilkan pesan "NIP/NIM telah digunakan".

Post-condition	Aktor telah berhasil menambahkan akun baru

ww) *Use Case Scenario* Melihat List Akun

Melihat list akun merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat daftar seluruh akun yang ada di sistem. *Use Case Scenario* melihat list akun dapat dilihat pada tabel berikut.

Tabel 4.53 *Use Case Scenario* Melihat List Akun

Flow of Events for Melihat List Akun	
Kode Kebutuhan	AME_F_4900
Objective	Untuk menampilkan seluruh list akun
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi
Main Flow	1. Aktor memilih menu Manajemen List Akun 2. Sistem membaca data akun di database 3. Sistem menampilkan seluruh list akun
Alternative Flow	-
Post-condition	Aktor dapat melihat seluruh list akun

xx) *Use Case Scenario* Mencari Akun

Mencari akun merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mencari akun dari daftar akun. *Use Case Scenario* mencari akun dapat dilihat pada tabel berikut.

Tabel 4.54 *Use Case Scenario* Mencari Akun

Flow of Events for Mencari Akun	
Kode Kebutuhan	AME_F_5000
Objective	Untuk mencari akun
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen List Akun
Main Flow	1. Aktor mengisi *field* pencarian akun 2. Sistem menampilkan akun yang dicari aktor
Alternative Flow	1. Jika akun yang dicari aktor tidak ditemukan maka aktor akan diberikan pesan "*No matching records found*"

Post-condition	Aktor dapat melihat akun yang dicari

yy) *Use Case Scenario* Mengedit Akun

Mengedit akun merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah data akun seperti username, nama lengkap dan lain-lain. *Use Case Scenario* mengedit akun dapat dilihat pada tabel berikut.

Tabel 4.55 *Use Case Scenario* Megedit Akun

Flow of Events for Mengedit	
Kode Kebutuhan	AME_F_5100
Objective	Untuk mengedit akun
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman edit akun
Main Flow	1. Aktor mengubah *field* NIM/NIP, nama unit, nama ketua unit, email, no. hp, *role* atau keterangan dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit akun 3. Sistem mengedit data akun di database 4. Sistem menampilkan halaman edit akun dan menampilkan notifikasi berisi pesan "Berhasil mengedit akun"
Alternative Flow	1. Jika *form* edit akun tidak valid maka sistem akan menampilkan notifikasi berisi pesan *error* 2. Jika aktor mengganti NIM/NIP dan NIM/NIP yang dimasukkan telah dipakai aktor lain maka sistem akan menampilkan pesan "NIP/NIM telah digunakan"
Post-condition	Aktor berhasil mengedit akun

zz) *Use Case Scenario* Menghapus Akun

Menghapus akun merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menghapus akun dari sistem. *Use Case Scenario* menghapus akun dapat dilihat pada tabel berikut.

Tabel 4.56 *Use Case Scenario* Menghapus Akun

Flow of Events for Menghapus Akun	
Kode Kebutuhan	AME_F_5200

Objective	Untuk menghapus akun
Aktor	Super Admin
Pre-condition	1. Aktor terauthentikasi dan terauthorisasi 2. Aktor mengakses halaman Manajemen List Akun
Main Flow	1. Aktor menekan tombol hapus 2. Sistem menghapus akun dari database 3. Sistem menampilkan halaman manajemen list akun dan menampilkan notifikasi berisi pesan "Berhasil menghapus akun"
Alternative Flow	1. Jika aktor menekan tombol batal pada alert konfirmasi penghapusan, maka proses penghapusan akan dibatalkan.
Post-condition	Aktor telah berhasil menghapus akun

aaa) *Use Case Scenario* Mereset Password

Mereset Password merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengganti password akunnya disebabkan lupa atau hal lainnya. *Use Case Scenario* Mereset Password dapat dilihat pada tabel berikut.

Tabel 4.57 *Use Case Scenario* Mereset Password

Flow of Events for Mereset Password	
Kode Kebutuhan	AME_F_5300
Objective	Untuk mereset password akun
Aktor	Super Admin
Pre-condition	1. Aktor terauthentikasi dan terauthorisasi 2. Aktor mengakses halaman Manajemen List Akun
Main Flow	1. Aktor menekan tombol *reset password* 2. Sistem mengupdate *field password* di *database* 3. Sistem mengirimkan email kemudian menampilkan halaman manajemen *list* akun dan notifikasi berisi pesan "Berhasil mereset *password* akun"
Alternative Flow	1. Jika aktor menekan tombol batal pada alert konfirmasi reset *password*, maka proses *reset password* akan dibatalkan.
Post-condition	Aktor telah berhasil mereset *password* akun

bbb) *Use Case Scenario* Melihat Seluruh List Peminjaman Venue

Melihat Seluruh List Peminjaman Venue merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat seluruh list peminjaman venue. *Use Case Scenario* Melihat Seluruh List Peminjaman Venue dapat dilihat pada tabel berikut.

Tabel 4.58 *Use Case Scenario* Melihat Seluruh List Peminjaman Venue

Flow of Events for Melihat Seluruh List Peminjaman Venue	
Kode Kebutuhan	AME_F_5400
Objective	Untuk menampilkan seluruh list peminjaman *venue*
Aktor	Super Admin, TU Subbag Umum dan Perlengkapan
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi
Main Flow	1. Aktor menekan mengakses halaman list peminjaman *venue* 2. Sistem membaca data peminjaman *venue* dari *database* 3. Sistem menampilkan seluruh *list* peminjaman *venue*
Alternative Flow	1. Jika belum ada list peminjaman venue, maka sistem akan menampilkan pesan "*No data available in table*"
Post-condition	Aktor dapat melihat seluruh *list* peminjaman *venue*

ccc) *Use Case Scenario* Menghapus Peminjaman Venue

Menghapus Peminjaman Venue merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menghapus peminjaman venue. *Use Case Scenario* Menghapus Peminjaman Venue dapat dilihat pada tabel berikut.

Tabel 4.59 *Use Case Scenario* Menghapus Peminjaman Venue

Flow of Events for Menghapus Peminjaman Venue	
Kode Kebutuhan	AME_F_5500
Objective	Untuk menghapus peminjaman *venue*
Aktor	TU Subbag Umum dan Perlengkapan, TU Subbag Akademik, Super Admin
Pre-condition	1. Aktor telah terautentikasi dan terautorisasi 2. Aktor telah berada di halaman list peminjaman venue
Main Flow	1. Aktor menekan *button* untuk menghapus peminjaman *venue* 2. Sistem menghapus peminjaman *venue* dari

	database 3. Sistem menampilkan halaman *list* peminjaman *venue* dan menampilkan notifikasi berisi pesan "Berhasil menghapus peminjaman *venue*"
Alternative Flow	1. Jika aktor menekan tombol "batal" pada alert konfirmasi maka penghapusan peminjaman *venue* akan dibatalkan 2. Perluasan ke *use case* menghapus *event* di Google Calendar.
Post-condition	Aktor telah berhasil menghapus peminjaman *venue*

ddd) *Use Case Scenario* Menghapus *Event*

Menghapus *event* merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menghapus *event*. *Use Case Scenario* menghapus *event* dapat dilihat pada tabel berikut.

Tabel 4.60 *Use Case Scenario* Menghapus *Event*

Flow of Events for Menghapus *Event*	
Kode Kebutuhan	AME_F_5600
Objective	Untuk menghapus *event*
Aktor	Super Admin
Pre-condition	1. Aktor telah terauthentikasi dan terauthorisasi 2. Aktor telah mengakses halaman Manajemen *List Event*
Main Flow	1. Aktor menekan tombol hapus 2. Sistem menghapus data *event* dari database 3. Sistem menampilkan halaman manajemen list *event* dan menampilkan notifikasi berisi pesan "Berhasil menghapus *event*".
Alternative Flow	1. Jika aktor menekan tombol batal pada alert konfirmasi maka proses penghapusan akan dibatalkan. 2. Jika data peminjaman *venue* dari *event* yang dihapus telah dimasukkan kedalam Google Calendar, maka sistem akan menghapus data peminjaman *venue* dari *event* tersebut di Google Calendar.
Post-condition	Aktor telah berhasil menghapus *event*

eee) *Use Case Scenario* Mengedit Data Event

Mengedit data event merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk mengubah data event seperti judul kegiatan, nama event dan lain-lain. *Use Case Scenario* mengedit data event dapat dilihat pada tabel berikut.

Tabel 4.61 Use Case Scenario Mengedit Data Event

Flow of Events for Mengedit Data Event	
Kode Kebutuhan	AME_F_5700
Objective	Untuk mengubah data *event*
Aktor	Super Admin
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi 2. Aktor telah berada di halaman edit event
Main Flow	1. Aktor mengisi field judul kegiatan atau deskripsi *event* dan menekan tombol simpan perubahan 2. Sistem melakukan validasi terhadap *form* edit *event* 3. Sistem menyimpan perubahan kedalam *database* 4. Sistem menampilkan halaman edit event dan menampilkan notifikasi berisi pesan "Berhasil mengedit data *event*"
Alternative Flow	1. Jika form edit data event tidak valid maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor berhasil mengedit data *event*

fff) *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Akademik

Menyetujui Peminjaman *Venue* dari TU Akademik merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk menyetujui peminjaman venue. *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Akademik dapat dilihat pada tabel berikut.

Tabel 4.62 *Use Case Scenario* Menyetujui Peminjaman *Venue* dari TU Akademik

Flow of Events for Menyetujui Peminjaman *Venue* dari TU Akademik	
Kode Kebutuhan	AME_F_5800
Objective	Untuk menyetujui peminjaman venue penyelenggaraan event
Aktor	TU Subbag Akademik
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi. 2. Aktor telah berada di halaman persetujuan event.

Main Flow	1. Aktor mengisi *field* catatan dan menekan tombol setujui 2. Sistem melakukan validasi *form* persetujuan peminjaman *venue* 3. Sistem mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan berhasil "Berhasil menyetujui peminjaman *venue*"
Alternative Flow	1. Perluasan ke use case memasukkan event ke Google Calendar. 2. Jika form edit persetujuan peminjaman venue tidak valid maka sistem akan menampilkan notifikasi berisi pesan *error*.
Post-condition	Aktor telah berhasil menyetujui peminjaman venue

ggg) *Use Case Scenario* Menampilkan Detail Akun

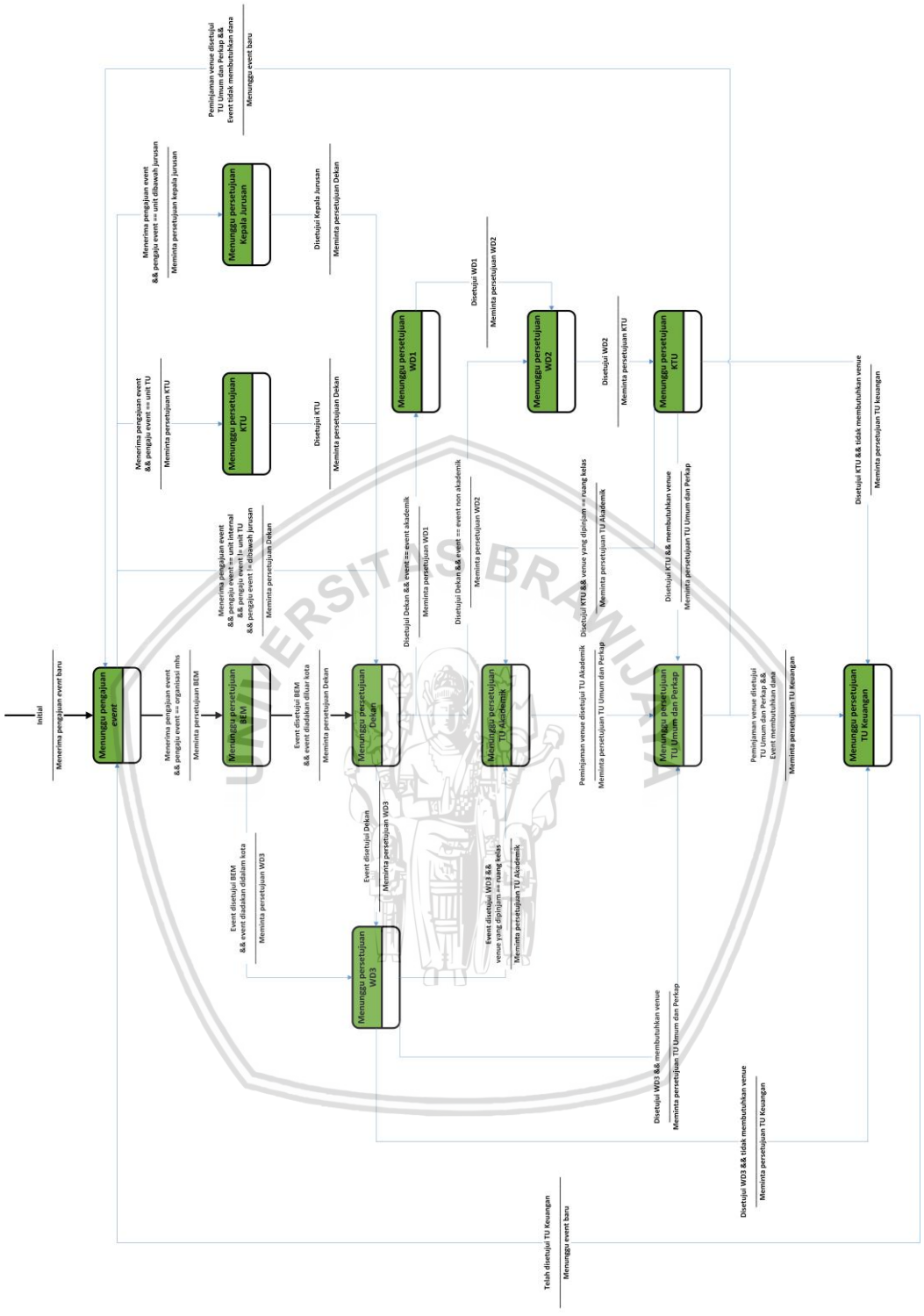
Menampilkan Detail Akun merupakan fungsionalitas yang disediakan sistem untuk memfasilitasi aktor untuk melihat detail akun. *Use Case Scenario* Menampilkan Detail Akun dapat dilihat pada tabel berikut.

Tabel 4.63 *Use Case Scenario* Menampilkan Detail Akun

Flow of Events for Menampilkan Detail Akun	
Kode Kebutuhan	AME_F_5900
Objective	Untuk melihat detail akun
Aktor	Super Admin
Pre-condition	1. Aktor telah terautentikasi dan terauthorisasi.
Main Flow	1. Aktor mengakses halaman detail akun 2. Sistem menampilkan halaman detail akun
Alternative Flow	-
Post-condition	Aktor melihat halaman detail akun

4.7.3 State Transition Diagram (STD)

State Transition Diagram (STD) merupakan salah satu diagram UML yang digunakan untuk menggambarkan perubahan *state* sistem. Pada penelitian ini, STD digunakan untuk memodelkan alur dan status disposisi proposal atau persetujuan *event*. Gambar 4.10 dibawah ini merupakan *State Transition Diagram* dari penelitian ini.



Gambar 4.10 *State Transition Diagram* (STD)

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan membahas proses perancangan dan implementasi aplikasi Manajemen *Event*. Karena pada penelitian ini penulis menggunakan pendekatan *Object Oriented*, maka pada tahap perancangan penulis akan menggunakan metode OOD (*Object Oriented Design*) dan pada tahap implementasi penulis akan menggunakan metode OOP (*Object Oriented Programming*).

Perancangan sistem akan dibagi menjadi 4 sub bab yaitu perancangan arsitektur yang berisi *Class Diagram* dan *Sequence Diagram* kemudian perancangan komponen yang berisi *pseudocode* dilanjutkan dengan perancangan data yang berisi *Entity Relationship Diagram* dan perancangan antarmuka dari beberapa halaman utama dari aplikasi ini. Pada proses implementasi terdapat 4 sub bab yang berisi hasil pengimplementasian dari perancangan yang telah dilakukan sebelumnya.

5.1 Perancangan Sistem

Setelah proses analisis kebutuhan telah dilakukan menggunakan metode OOA (*Object Oriented Analysis*), proses yang selanjutnya yaitu OOD (*Object Oriented Design*). OOD merupakan suatu proses perancangan suatu sistem atau aplikasi dengan menggunakan pendekatan berorientasi objek. OOD menggunakan hasil dari fase OOA sebagai *input*. Pada penelitian ini proses perancangan akan dibagi menjadi 4 yaitu perancangan arsitektur, perancangan komponen, perancangan data dan perancangan antarmuka (*user interface*).

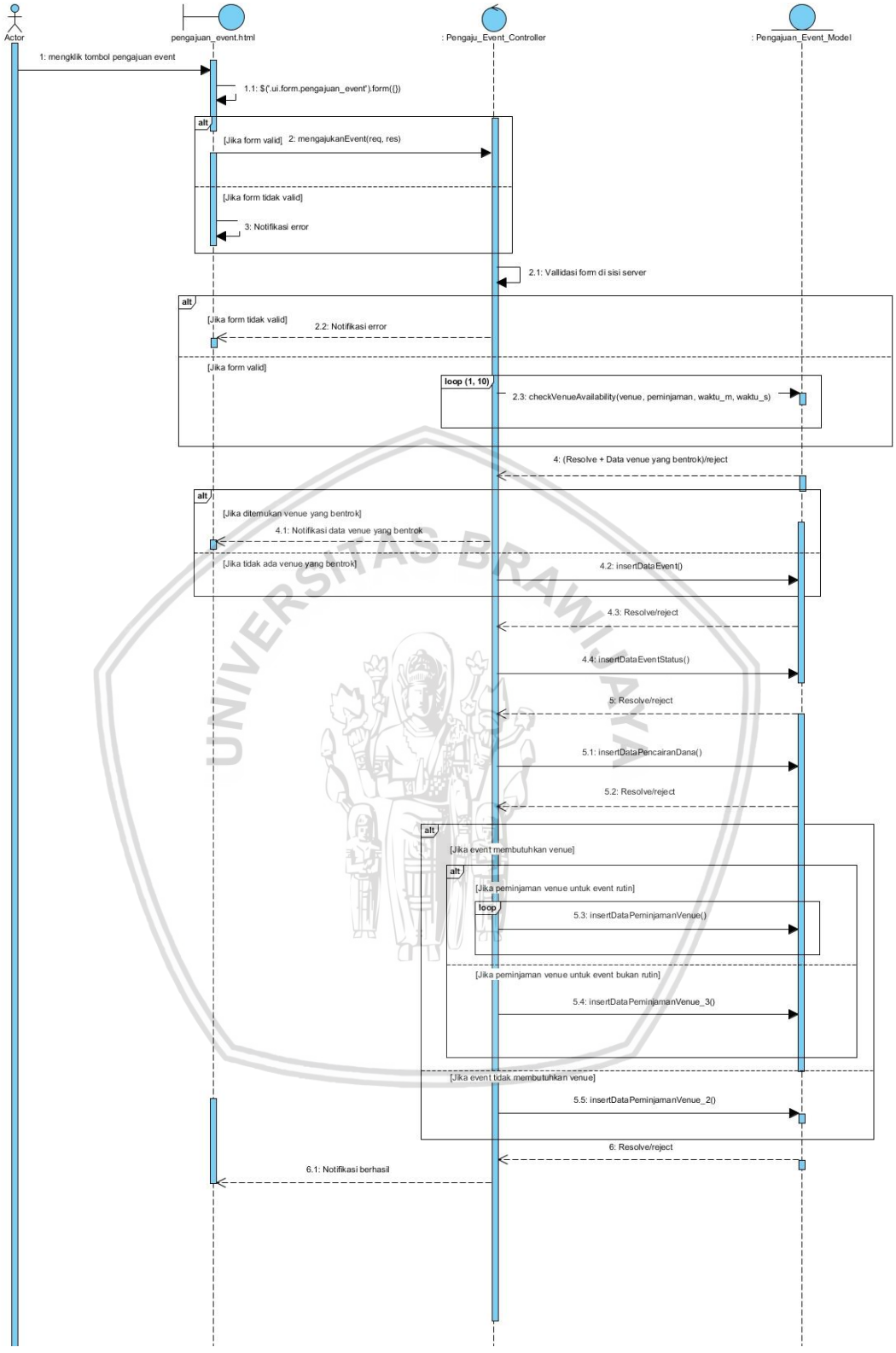
5.1.1 Perancangan Arsitektur

Pada bab Perancangan Arsitektur ini akan dibahas secara terperinci mengenai *Sequence Diagram* dan *Class Diagram* pada aplikasi ini. *Sequence Diagram* merupakan diagram yang menjelaskan interaksi antar objek. Sedangkan *Class Diagram* menjelaskan struktur kelas (atribut dan *method*) dan relasi antar kelas.

5.1.1.1 Sequence Diagram

a. Sequence Diagram Mengajukan *Event*

Gambar 5.1 dibawah ini merupakan *sequence diagram* dari fungsionalitas Mengajukan *Event*. Aktor di *sequence diagram* Mengajukan *Event* adalah seluruh aktor yang terdapat di aplikasi ini, kecuali non member dan super admin. Pada *sequence diagram* Mengajukan *Event* terdapat 1 *boundary* yaitu pengajuan_event.html. Terdapat satu *control* yaitu Pengaju_Event_Controller dan satu *entity* yaitu Pengajuan_Event_Model. Pada sequence diagram Mengajukan Event terdapat 5 *alternative fragment* dan 2 *loop fragment*.

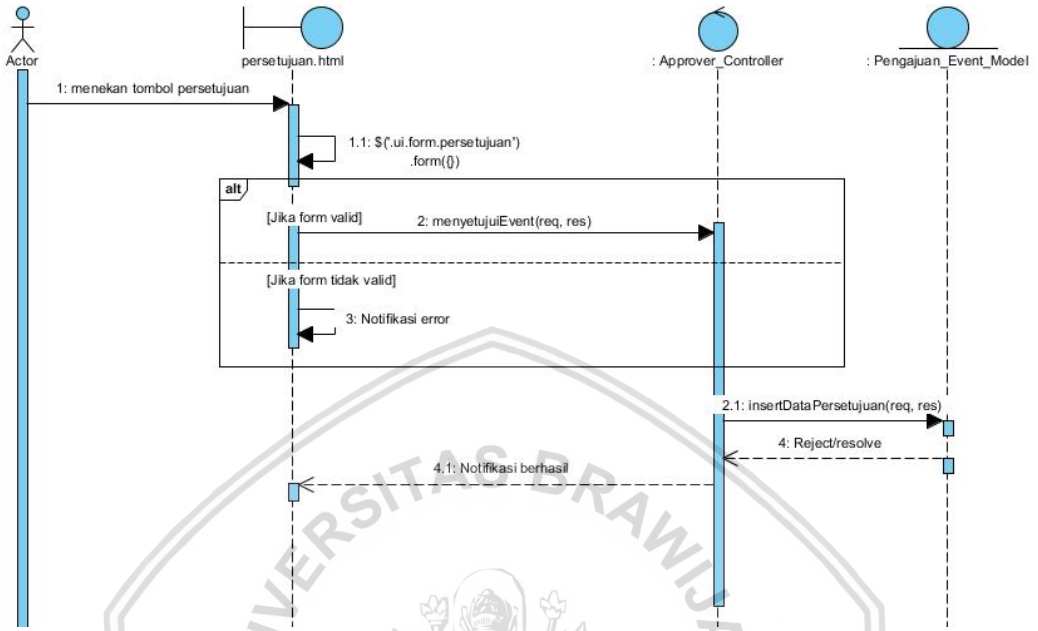


Gambar 5.1 *Sequence Diagram* Mengajukan *Event*

b. Sequence Diagram Menyetujui *Event*

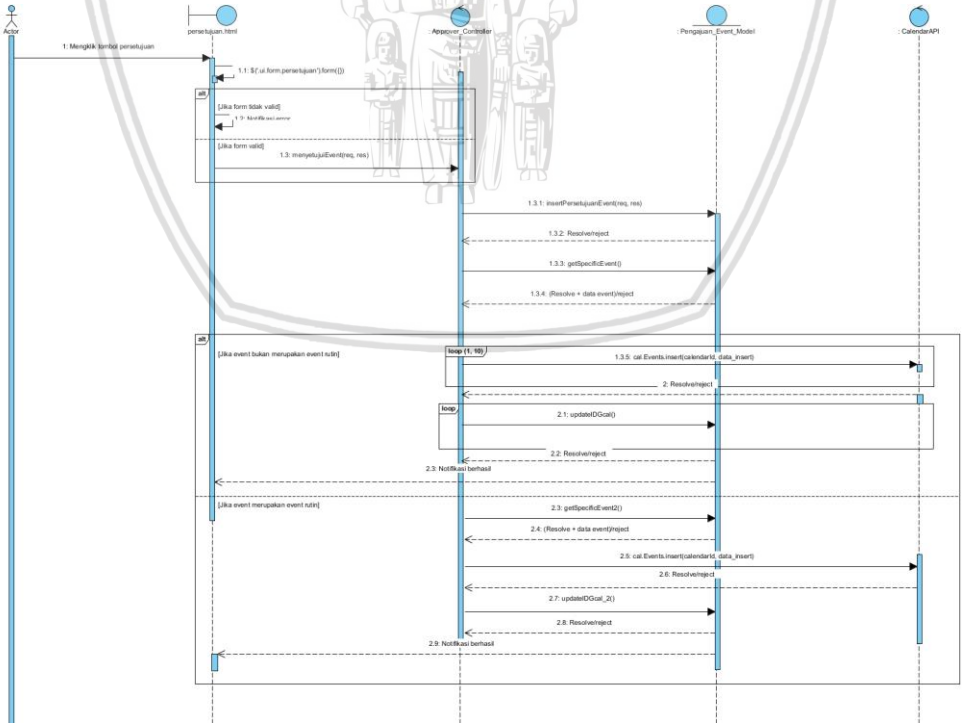
Gambar 5.2 dibawah ini merupakan *sequence diagram* dari fungsionalitas Menyetujui *Event*. Aktor di *sequence diagram* Menyetujui *Event* adalah Dekan,

KTU, WD1, Kajur dan BEM. Pada *sequence diagram* Menyetujui *Event* terdapat 1 *boundary* yaitu pengajuan_event.html. Terdapat satu *control* yaitu Approver_Controller dan satu *entity* yaitu Pengajuan_Event_Model. Pada *sequence diagram* Menyetujui *Event*, terdapat 1 *alternative fragment*.



Gambar 5.2 *Sequence Diagram* Menyetujui *Event*

c. *Sequence Diagram* Menyetujui Peminjaman Venue dari TU Umum Perkap



Gambar 5.3 *Sequence Diagram* Menyetujui Peminjaman *Venue* dari TU Umum Perkap

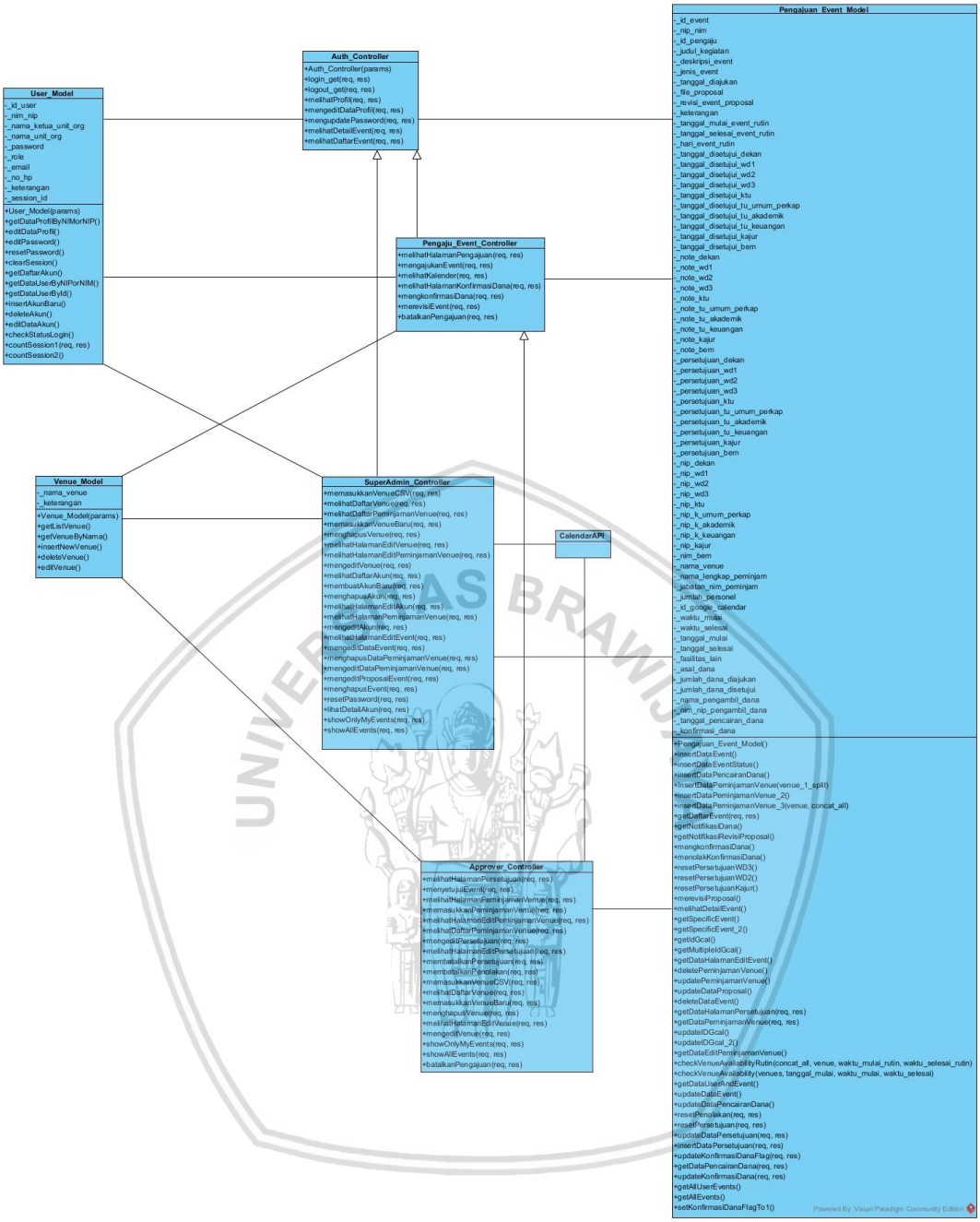
Gambar 5.3 dibawah ini merupakan *sequence diagram* dari fungsionalitas Menyetujui Peminjaman *Venue* dari TU Umum Perkap. Aktor di *sequence diagram* Menyetujui Peminjaman *Venue* dari TU Umum Perkap adalah TU Subbag Umum dan Perlengkapan. Pada *sequence diagram* ini terdapat 1 *boundary* yaitu pengajuan_event.html. Terdapat dua *control* yaitu Approver_Controller dan CalendarAPI (*controller* dari aplikasi Google Calendar) dan satu *entity* yaitu Pengajuan_Event_Model. Pada *sequence diagram* Menyetujui Peminjaman Venue dari TU Umum Perkap terdapat 2 *alternative fragment* dan 2 *loop fragment*.

5.1.1.2 Class Diagram

Class diagram merupakan diagram yang menggambarkan struktur kelas (*property* dan *method*) dan relasi antar kelas. Pada OOD, *candidate class* didapat dari fase OOA (*Object Oriented Analysis*). Untuk mengidentifikasi *candidate class* pada fase OOA, penulis melakukan analisis terhadap *noun* (kata benda) dan *pronoun* (kata benda pengganti) yang ada pada *main flow* dan *alternative flow* yang terdapat di *use case scenario*.

Kemudian setelah penulis berhasil mengidentifikasi *candidate class*, penulis selanjutnya menentukan kelas-kelas mana saja yang akan diimplementasikan dari *candidate class* yang ada. Penulis telah menentukan terdapat 8 kelas pada aplikasi Manajemen *Event* yang terbagi menjadi menjadi Controllers (Auth_Controller, Pengaju_Event_Controller, Approver_Controller, SuperAdmin_Controller) dan Models (Pengajuan_Event_Model, Venue_Model, User_Model) serta kelas CalendarAPI (kelas dari Google Calendar API) yang diilustrasikan pada Gambar 5.4 dibawah.

Pada class diagram di penelitian ini terdapat 2 jenis relasi yaitu *association* dan *generalization.* Approver_Controller merupakan *subclass* dari kelas Pengaju_Event_Controller. Pengaju_Event_Controller dan SuperAdmin_ Controller merupakan *subclass* dari Auth_Controller. Relasi antar kelas yang lainnya adalah *association.* Setiap atribut pada kelas model memiliki *prefix underscore* (_) yang merupakan *naming convention* yang menunjukkan bahwa variabel tersebut harus diperlakukan sebagai *private*.



Gambar 5.4 *Class Diagram*

5.1.2 Perancangan Komponen

Pada perancangan komponen penulis akan menjelaskan algoritma dari beberapa *method* yang akan menjadi dasar pada tahap implementasi kode program.

5.1.2.1 Perancangan Komponen *Method* Melihat Halaman Persetujuan

Nama Kelas: Approver_Controller

Nama Operasi: `melihatHalamanPersetujuan()`

Deskripsi : *Pseudocode* ini berfungsi untuk melihat halaman persetujuan *event.*

Algoritma :

Tabel 5.1 Perancangan Komponen *Method* Melihat Halaman Persetujuan

No	Perancangan Komponen *Method* Melihat Halaman Persetujuan
1	`Deklarasi var pengajuan_event`
2	`Deklarasi var params`
3	`Deklarasi var total`
4	`Pengajuan_event= new Pengajuan_Event(params)`
5	`Panggil method getDataHalamanPersetujuan()`
6	`If (result > 0)`
7	`Session notif <- result.length`
8	`Render halaman persetujuan`
9	`Else`
10	`Generate notifikasi ”Belum ada event”`
11	`Render halaman persetujuan`
12	`Endif`

5.1.2.2 Perancangan Komponen *Method* Mengkonfirmasi Dana

Nama Kelas: Pengaju_Event_Controller

Nama Operasi: `mengkonfirmasiDana()`

Deskripsi : *Pseudocode* ini berfungsi untuk mengkonfirmasi atau menolak dana yang disetujui oleh *approver*.

Algoritma :

Tabel 5.2 Perancangan Komponen *Method* Mengkonfirmasi Dana

No	Perancangan Komponen *Method* Mengkonfirmasi Dana
1	`Deklarasi var pengajuan_event`
2	`Params <- {id_event : ID event yang ada di URL}`
3	`Deklarasi var total`
4	`Pengajuan event = new Pengajuan_Event_Controller(params)`
5	`If(button konfirmasi dana diklik)`
6	`Panggil method mengkonfirmasiDana menggunakan objek`
7	`pengajuan_event`
8	`Decrement session yang menyimpan jumlah notifikasi`
9	`Generate pesan notifikasi berhasil`
10	`Arahkan aktor ke halaman pengajuan event`
11	`Elif(button tolak dana diklik)`
12	`Panggil method menolakKonfirmasiDana menggunakan`

```
13        objek pengajuan_event
14        Panggil method getDataPencairanDana menggunakan objek
15        pengajuan_event
16        if(asal dana adalah fakultas/kemahasiswaan dan pengaju
17        event adalah organisasi mahasiwa)
18              panggil method resetpersetujuanWD3
19        elif(asal dana adalah fakultas/kemahasiswaan dan
20        pengaju event adalah unit internal)
21              panggil method resetPersetujuanWD2
22        Endif
23        Decrement session yang menyimpan jumlah notifikasi
24        Generate pesan berhasil
25        Arahkan aktor ke halaman pengajuan event
26  Endif
```

5.1.2.3 Perancangan Komponen *Method* Memasukkan *Venue* Baru

Nama Kelas: SuperAdmin_Controller

Nama Operasi: `memasukkanVenueBaru()`

Deskripsi : *Pseudocode* ini berfungsi untuk memasukkan *venue* baru ke list *venue*.

Algoritma :

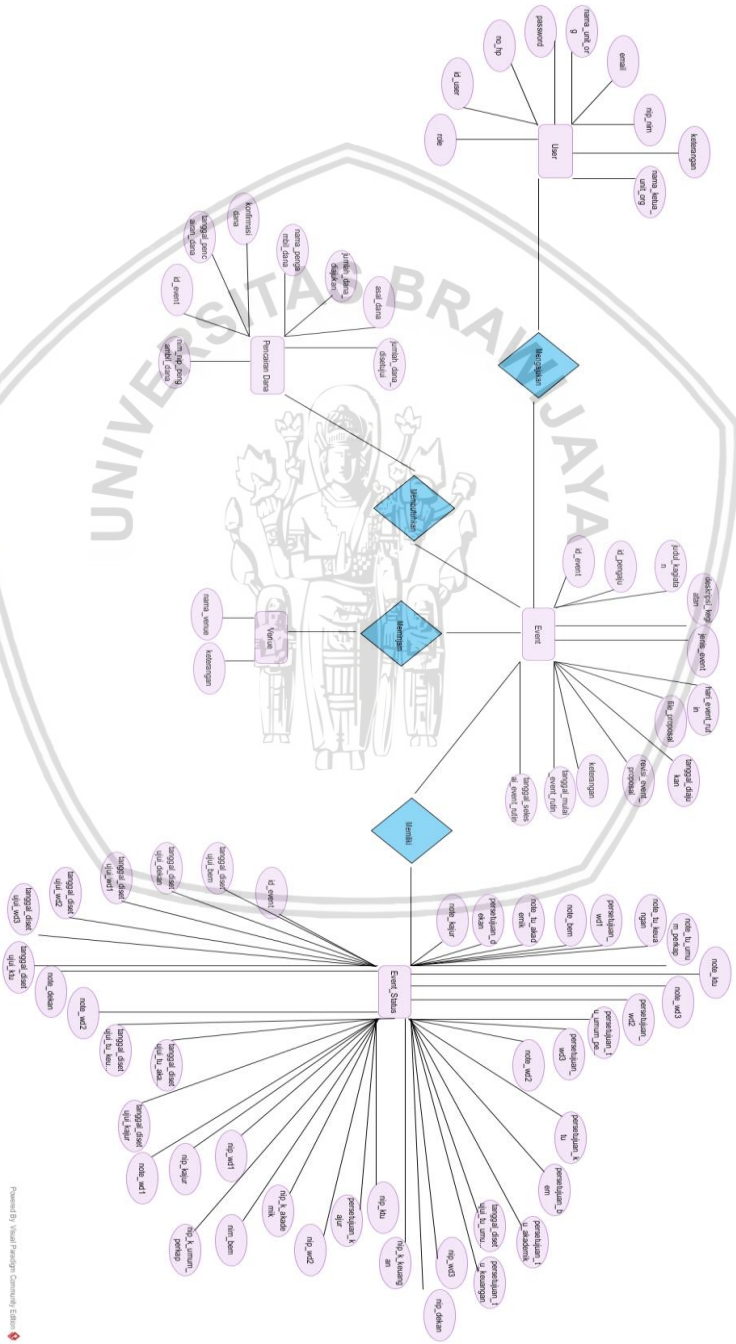
Tabel 5.3 Perancangan Komponen *Method* Memasukkan *Venue* Baru

No	Perancangan Komponen *Method* Memasukkan *Venue* Baru
1	`Nama venue <- nama venue yang diinputkan user`
2	`Keterangan <- keterangan yang diinputkan user`
3	`Params <- {nama venue: nama venue,`
4	`Keterangan : keterangan}`
5	`venue = new Venue_Model(params)`
6	`If(exist(nama_venue)&& exist(keterangan)`
7	`Panggil method getVenueByNama`
8	`If(exist(result))`
9	`Generate pesan error venue telah ada di db`
10	`Arahkan aktor ke halaman List Venue`
11	`Else`
12	`Panggil method insertNewVenue`
13	`Generate pesan sukses`
14	`Arahkan aktor ke halaman List Venue`
15	`Endif`

16	Else
17	Generate pesan error
18	Arahkan aktor ke halaman List Venue
19	Endif

5.1.3 Perancangan Data

5.1.3.1 Entity Relationship Diagram (ERD)



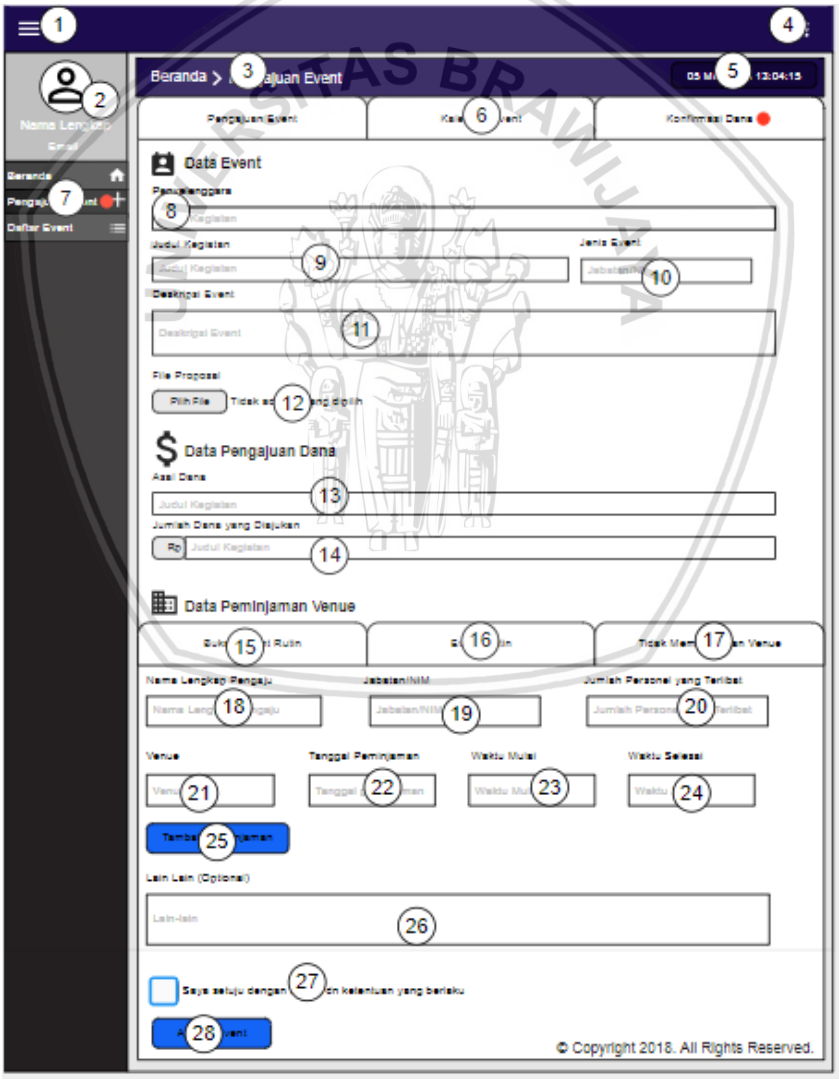
Gambar 5.5 *Entity Relationship Diagram*

Gambar 5.5 diatas merupakan *entity relationship diagram* pada penelitian ini. Terdapat 5 entitas yang teridentifikasi yaitu user, venue, event, event status dan pencairan dana. Entitas user dan event dihubungkan dengan relasi bernama mengajukan. Pencairan dana dan event dihubungkan dengan relasi bernama membutuhkan. Event dan venue dihubungkan dengan relasi bernama meminjam. Event dan event status dihubungkan dengan relasi bernama memiliki.

5.1.4 Perancangan Antarmuka

Perancangan antarmuka dilakukan dengan membuat *mock-up* dari beberapa halaman yang ada pada aplikasi Manajemen *Event.* Hasil pada tahap ini digunakan sebagai acuan untuk implementasi antarmuka pada tahap implementasi perangkat lunak.

5.1.4.1 Perancangan Antarmuka Mengajukan *Event*



Gambar 5.6 Perancangan Antarmuka Pengajuan *Event*

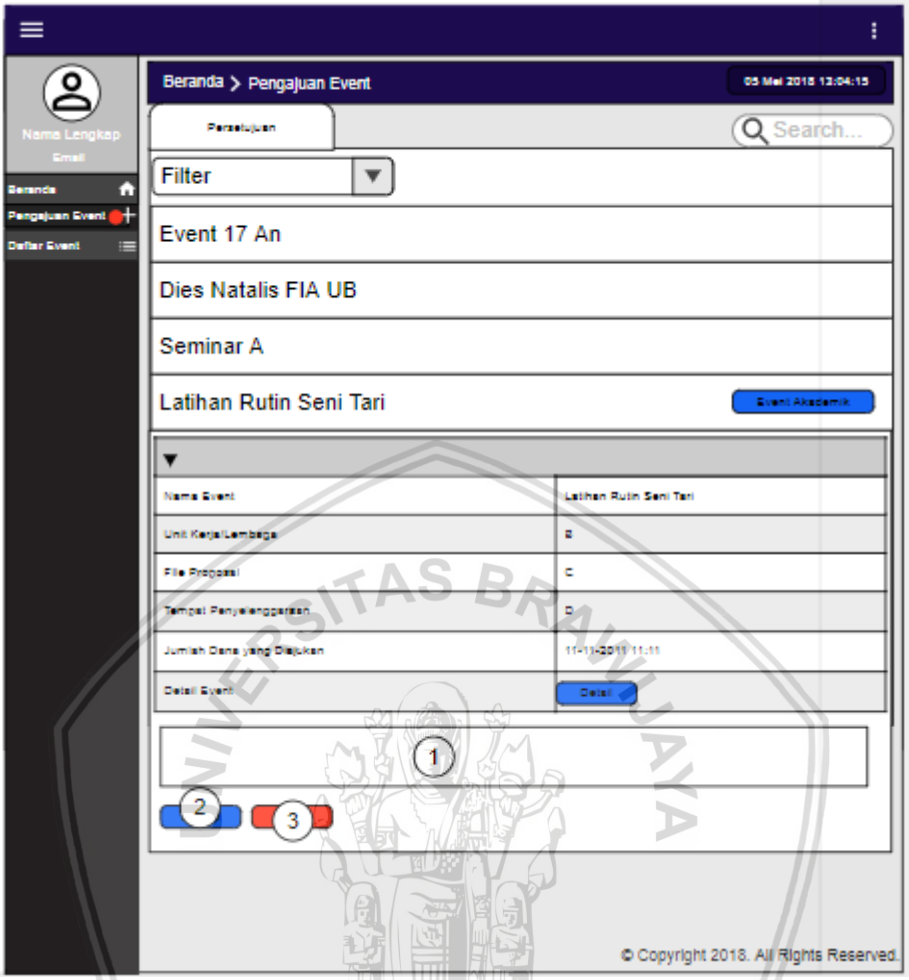
Gambar 5.6 diatas merupakan gambaran perancangan antarmuka untuk pengajuan *event*. Tabel 5.4 dibawah ini berisi penjelasan masing-masing komponen pada Gambar 5.6.

Tabel 5.4 Perancangan Antarmuka Pengajuan Event

No	Nama Objek	Tipe	Keterangan
1	*Trigger sidebar*	*Button*	Digunakan untuk menampilkan dan menyembunyikan *sidebar*
2	Informasi *user*	*Text*	Berisi inisial (gabungan huruf pertama dari *First Name* dan *Last Name* Organisasi/Unit), nama unit/organisasi dan email
3	*Breadcrumb*	*Breadcrumb*	Menampilkan lokasi halaman yang sedang diakses aktor
4	*Button* edit profil	*Button*	Jika diklik akan muncul *button* yang mengarahkan ke halaman edit profil dan *button* untuk *logout*
5	Tanggal dan waktu	*Label*	Menampilkan waktu sekarang
6	*Tab*	*Tab*	Untuk berpindah tab dari pengajuan *event* ke melihat kalender dan notifikasi event
7	*Sidebar*	*Sidebar*	Digunakan untuk menampilkan list menu
8	Penyelenggara	*Textbox*	Digunakan untuk menginputkan penyelenggara *event*
9	Judul kegiatan	*Textbox*	Digunakan untuk menginputkan judul kegiatan
10	Jenis *event*	*dropdown*	Digunakan untuk menginputkan jenis event
11	Deskripsi *event*	*Textarea*	Digunakan untuk menginputkan deksripsi event
12	File proposal	*Button*	Digunakan untuk mengupload file proposal
13	Jumlah dana diajukan	*Textbbox*	Digunakan untuk menginputkan jumlah dana diajukan
14	Asal dana	*Dropdown*	Digunakan untuk menginputkan asal dana
15	Tab peminjaman *venue* untuk *event* tidak	*Tab*	Digunakan untuk menginputkan peminjaman *venue* untuk *event* yang bukan merupakan *event* rutin

	rutin		
16	Tab peminjaman *venue* untuk *event* rutin	*Tab*	Digunakan untuk menginputkan peminjaman *venue* untuk event yang merupakan *event* rutin
17	Tab tidak memerlukan *venue*	*Tab*	Jika tidak memerlukan *venue* maka aktor diharuskan mengisi data di *tab* ini
18	Nama lengkap pengaju	*Textbox*	Digunakan untuk menginputkan nama lengkap pengaju
19	NIP/NIM	*Textbox*	Digunakan untuk menginputkan NIP/NIM
20	Jumlah personel yang terlibat	*Textbox*	Digunakan untuk menginputkan jumlah personel yang terlibat (panitia + peserta)
21	*Venue*	*Multiple selection dropdown*	Digunakan untuk menginputkan *venue* yang akan dipinjam. Bisa memilih lebih dari 1 *venue*.
22	Tanggal peminjaman	*Textbox*	Digunakan untuk menginputkan tanggal peminjaman *venue*
23	Waktu mulai	*Textbox*	Digunakan untuk menginputkan waktu mulai peminjaman *venue*
24	Waktu selesai	*Textbox*	Digunakan untuk menginputkan waktu selesai peminjaman *venue*
25	Tambah peminjaman	*Button*	Digunakan untuk menampilkan *field* peminjaman *venue* baru. Maksimal sistem dapat menampilkan 10 *field* peminjaman *venue*.
26	Lain lain	*Textarea*	Digunakan untuk menginputkan lain-lain
27	*Checkbox* S&K	*Checkbox*	Digunakan untuk menginputkan persetujuan terkait syarat dan ketentuan peminjaman *venue* di FIA UB
28	*Button* mengajukan *event*	*Button*	Digunakan untuk mengirimkan form pengajuan *event*

5.1.4.2 Perancangan Antarmuka Menyetujui Event



Gambar 5.7 Perancangan Antarmuka Menyetujui Event

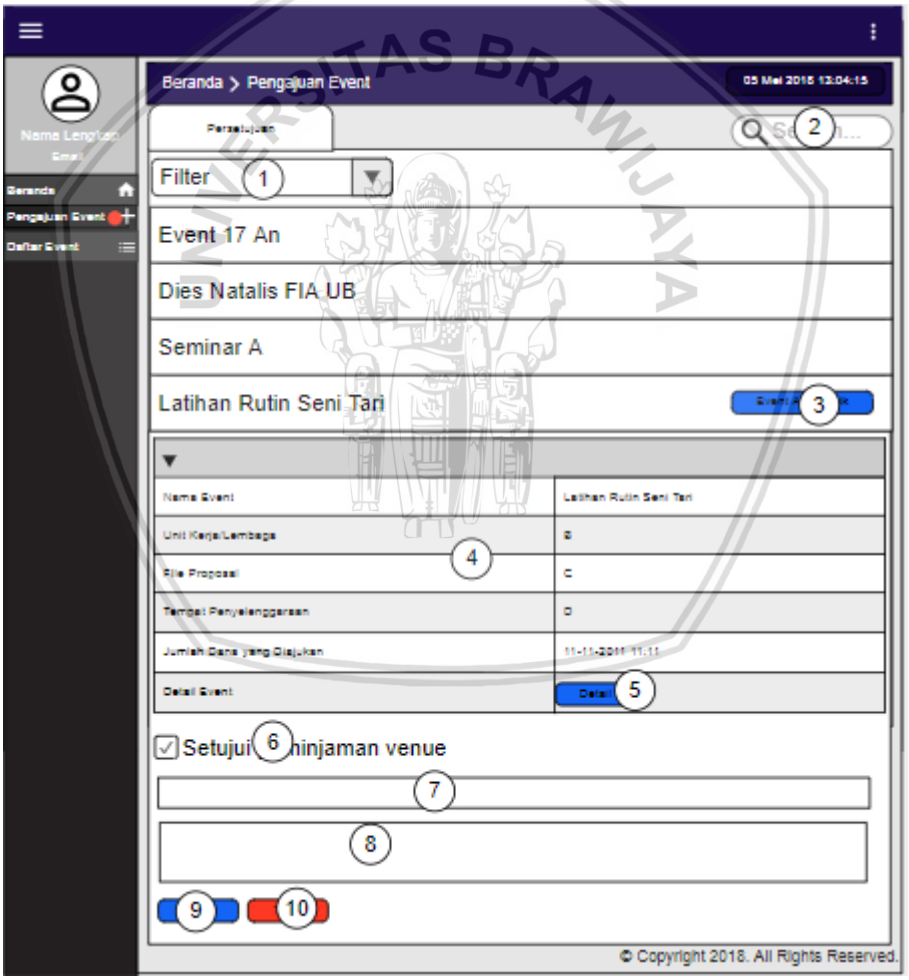
Gambar 5.7 diatas merupakan gambaran perancangan antarmuka untuk menyetujui event. Tabel 5.5 dibawah ini berisi penjelasan masing-masing komponen pada Gambar 5.7.

Tabel 5.5 Perancangan Antarmuka Pengajuan Event

No	Nama Objek	Tipe	Keterangan
1	Trigger sidebar	*Button*	Digunakan untuk menampilkan dan menyembunyikan sidebar
2	Informasi user	*Text*	Berisi inisial (gabungan huruf pertama dari First Name dan Last Name Organisasi/Unit), nama unit/organisasi dan email
3	Breadcrumb	*Breadcrumb*	Menampilkan lokasi halaman yang sedang diakses aktor
4	Button edit	*Button*	Jika diklik akan muncul button yang

	profil		mengarahkan ke halaman edit profil dan button untuk logout
5	Tanggal dan waktu	*Label*	Menampilkan waktu sekarang
6	Tab	*Tab*	Untuk berpindah tab dari pengajuan event ke melihat kalender dan notifikasi event
7	Sidebar	*Sidebar*	Digunakan untuk menampilkan list menu
8	Penyelenggara	*Textbox*	Digunakan untuk menginputkan penyelenggara event

5.1.4.3 Perancangan Antarmuka Menyetujui Event dari WD2 dan WD3



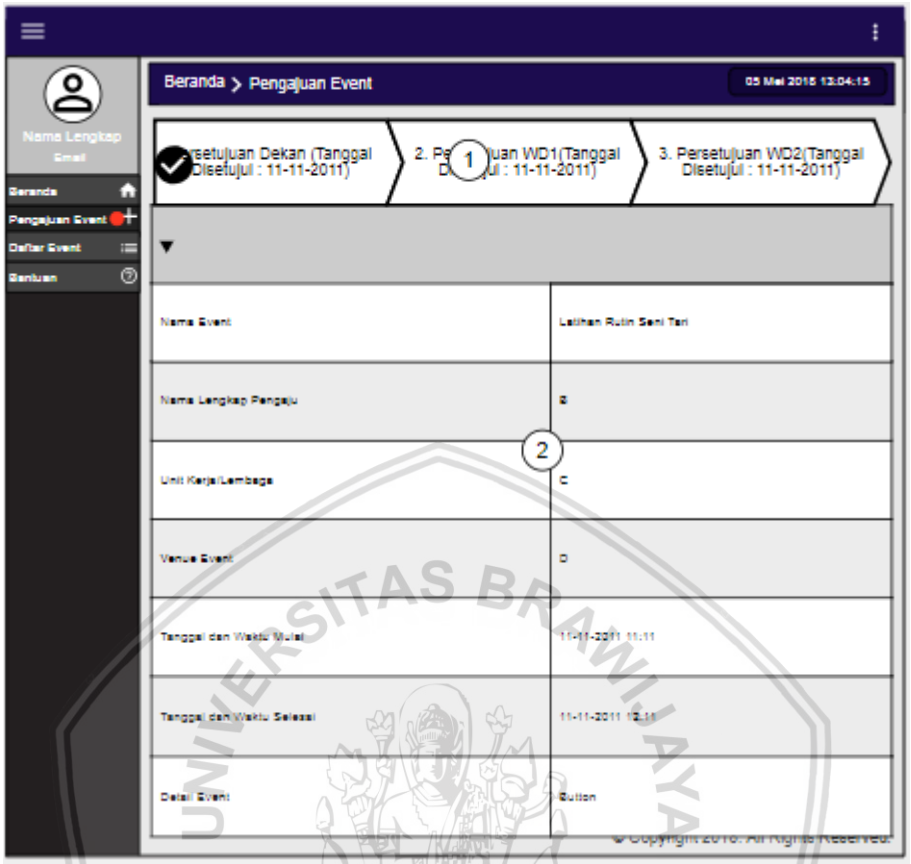
Gambar 5.8 Perancangan Antarmuka Menyetujui Event dari WD2 dan WD3

Gambar 5.8 diatas merupakan gambaran perancangan antarmuka untuk menyetujui event dari WD2 dan WD3. Tabel 5.6 dibawah ini berisi penjelasan masing-masing komponen pada Gambar 5.8.

Tabel 5.6 Perancangan Antarmuka Pengajuan Event

No	Nama Objek	Tipe	Keterangan
1	Filter	*Dropdown*	Menentukan berapa jumlah list venue yang ditampilkan di halaman awal
2	Search event	*Textbox*	Digunakan untuk mencari event
3	Jenis event	*Label*	Menampilkan jenis dari suatu event (akademik, non akademik atau event mahasiswa)
4	Tabel event	*Table*	Berisi informasi event seperti judul kegiatan, unit kerja/lembaga, link file proposal, tempat penyelenggaraan beserta waktunya, jumlah dana diajukan dan button detail event.
5	Button detail event	*Button*	Jika diklik akan mengarahkan aktor ke halaman detail event
6	Checkbox perizinan peminjaman venue	*Checkbox*	Digunakan oleh aktor untuk mengizinkan pengajuan peminjaman venue
7	Dana yang disetujui	*Textbox*	Digunakan untuk memasukkan jumlah dana yang disetujui oleh approver
8	Catatan	*Textarea*	Digunakan untuk menginputkan catatan dari aktor
9	Button setujui	*Button*	Digunakan untuk menyetujui event
10	Button tolak	*Button*	Digunakan untuk menolak persetujuan event

5.1.4.4 Perancangan Antarmuka Melihat Detail Event



Gambar 5.9 Perancangan Antarmuka Melihat Detail Event

Gambar 5.9 diatas merupakan gambaran perancangan antarmuka untuk melihat detail event. Tabel 5.7 dibawah ini berisi penjelasan masing-masing komponen pada Gambar 5.9.

Tabel 5.7 Perancangan Antarmuka Pengajuan Event

No	Nama Objek	Tipe	Keterangan
1	Status pengajuan event	*Step*	Digunakan untuk menampilkan status pengajuan event. Berisi informasi suatu event sudah disetujui atau belum. Jika sudah disetujui, akan ditampilkan nama dan NIP approver serta tanggal disetujui
2	Tabel detail event	*Table*	Berisi seluruh data event yaitu unit/organisasi event, judul kegiatan, no telp, email, jenis event, deskripsi event, venue yang dipinjam beserta waktu peminjaman masing-masing venue, jumlah dana diajukan, tanggal diajukan, jumlah personel yang terlibat, file proposal, lain-lain, catatan dari seluruh

			Approver yang menyetujui event ini, dan informasi pencairan dana (nama pengambil dana, nim/nip pengambil dana, tanggal pencairan dana).

5.2 Implementasi Sistem

Tahap selanjutnya setelah perancangan sistem adalah implementasi sistem. Untuk antarmuka sistem (*front-end*) akan diimplementasikan menggunakan bahasa pemrograman HTML, CSS dan Javascript dengan menggunakan SemanticUI dan jQuery. Sedangkan untuk fungsionalitas sistem (*back-end*) akan diimplementasikan menggunakan Javascript (Node.js) dengan *framework* Express.

5.2.1 Spesifikasi Sistem

Subbab spesifikasi sistem menjelaskan tentang spesifikasi dari perangkat keras dan perangkat lunak yang digunakan dalam pengembangan aplikasi Manajemen *Event* ini. Untuk lebih jelasnya akan dibahas pada subbab berikut ini.

5.2.1.1 Spesifikasi Perangkat Keras

Tabel 5.8 dibawah ini merupakan spesifikasi perangkat keras yang digunakan didalam penelitian ini.

Tabel 5.8 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	*Processor* : Intel® Core™ Core i7-4700HQ – 3.4GHz *Memory* : 4 GB DDR3 *Harddisk* : 1TB

5.2.1.2 Spesifikasi Perangkat Lunak

Tabel 5.9 dibawah ini merupakan spesifikasi perangkat lunak yang digunakan didalam penelitian ini.

Tabel 5.9 Spesifikasi Perangkat Lunak

Nama Komponen	Nama Produk/Aplikasi yang Digunakan
Editor Dokumentasi	Microsoft Word 2016
Editor Diagram Perancangan	Visual Paradigm *Community Edition*, Microsoft Visio 2016
IDE untuk pemrograman	Visual Studio Code
Bahasa Pemrograman	HTML, Javascript, CSS

Framework Back-end	Express.js
Framework Front-end	SemanticUI
Library Front-end	jQuery
Package-Package Utama yang Digunakan	NPM, nodemon, node-google-calendar, passport.js, nodemailer, express-session, csv-mysql, multer, swig, helmet, express-mysql-session
DBMS	MySQL
Server DBMS	XAMPP
Browser	Google Chrome
Sistem Operasi	Windows 8.1 64 bit

5.2.2 Implementasi Kode Program

Implementasi kode program adalah penterjemahan *pseudocode* yang sebelumnya telah dibuat pada proses perancangan kedalam bahasa pemrograman yang spesifik. Pada penelitian ini penulis menggunakan Node.js dengan bantuan *framework* Express untuk mengimplementasikan *back-end* dari aplikasi ini.

5.2.2.1 Implementasi Kode Program *Method* Melihat Halaman Persetujuan

Nama *Class*: Approver_Controller

Nama Operasi : `melihatHalamanPersetujuan()`

Deskripsi : Pseudocode ini berfungsi untuk melihat halaman persetujuan event.

Source Code:

Tabel 5.10 Implementasi Kode Program Melihat Halaman Persetujuan

```
1   melihatHalamanPersetujuan(req, res) {
2           var pengajuan_event;
3           var params = {};
4           var total;
5           pengajuan_event = new Pengajuan_Event_Model(params);
6           pengajuan_event.getDataHalamanPersetujuan(req, res)
7               .then(function (result) {
8                   if (result.length > 0) {
9                       req.session.notif = result.length;
10
11  res.render('pengajuan_event/persetujuan', {
12                          events: result,
```

```
                            person: req.user,
                            notif: req.session.notif,
                        })
                    }
                    else {
                        req.flash('message_notf', "Belum ada
event");

res.render('pengajuan_event/persetujuan', {
                            person: req.user,
                            message_err:
req.flash('message_err'),
                            message_success:
req.flash('message_success'),
                            message_notf:
req.flash('message_notf')
                        });
                }
            })
            .catch(function (err) {
                req.flash('message_err', "Internal server
error");
                res.redirect('/');
            });
    }
```

5.2.2.2 Implementasi Kode Program *Method* Mengkonfirmasi Dana

Nama *Class*: Pengaju_Event_Controller

Nama Operasi : `mengkonfirmasiDana()`

Deskripsi : Pseudocode ini berfungsi untuk mengkonfirmasi atau menolak dana yang disetujui oleh approver

Source Code:

Tabel 5.11 Implementasi Kode Program *Method* Mengkonfirmasi Dana

```
mengkonfirmasiDana(req, res) {
        var pengajuan_event;
        var params = { id_event: req.params.eventID };
        var total;
        pengajuan_event = new
Pengajuan_Event_Model(params);
```

```
7           if (helper.isExist(req.body.konfirmasi_dana)) {
8               pengajuan_event.mengkonfirmasiDana()
9                   .then(function () {
10                      req.session.notif_2--;
11                      req.flash('message_success', 'Berhasil
12  menyetujui dana event');
13                      res.redirect('/pengajuan_event');
14                  }).catch(function (err) {
15                      req.flash('message_err', "Internal
16  server error");
17                      res.redirect('/');
18                  }
19                  );
20          }
21          else if
22  (helper.isExist(req.body.tolak_konfirmasi_dana)) {
23              params = {
24                  id_event: req.params.eventID
25              };
26              pengajuan_event = new
27  Pengajuan_Event_Model(params);
28              pengajuan_event.menolakKonfirmasiDana()
29                  .then(function () {
30                      return
31  pengajuan_event.getDataPencairanDana()
32                  })
33                  .then(function (result) {
34                      if (result.asal_dana == 'fakultas_kmhs'
35  && (result.role == 'pengajuevent_mahasiswa' || result.role
36  == 'bem_fia')) {
37                          return
38  pengajuan_event.resetPersetujuanWD3();
39                      }
40                      else if (result.asal_dana ==
41  'fakultas_kmhs' && result.role !== 'pengajuevent_mahasiswa'
42  && result.role !== 'bem_fia') {
43                          return
44  pengajuan_event.resetPersetujuanWD2();
```

```
45                      }
46                  })
47                  .then(function () {
48                      req.session.notif_2--;
49                      req.flash('message_success', 'Berhasil
50  menolak dana event');
51                      res.redirect('/pengajuan_event');
52                  })
53                  .catch(function (err) {
54                      console.log(err);
55                      req.flash('message_err', "Internal
56  server error");
57                      res.redirect('/');
58                  })
59          }
60      }
```

5.2.2.3 Implementasi Kode Program *Method* Memasukkan Venue Baru

Nama *Class*: SuperAdmin_Controller

Nama Operasi : `memasukkanVenueBaru()`

Deskripsi : Pseudocode ini berfungsi untuk memasukkan venue baru ke list venue

Source Code:

Tabel 5.12 Implementasi Kode Program *Method* Memasukkan Venue Baru

```
1   memasukkanVenuBaru(req, res) {
2           var nama_venue = req.body.nama_venue;
3           var keterangan = req.body.keterangan;
4           var params = {
5               nama_venue: nama_venue,
6               keterangan: keterangan
7           }
8           var venue = new Venue_Model(params);
9           if (helper.isExist(nama_venue) &&
10  helper.isExist(keterangan)) {
11              venue.getVenueByNama()
12                  .then(function (result) {
13                      if (result) {
14                          req.flash('message_err', "Venue
```

```
15  dengan nama " + nama_venue + " telah ada didalam
16  database!");
17  res.redirect('/manajemen_list_venue');
18                      }
19                      else {
20                          venue.insertNewVenue()
21                  .then(function () {
22  req.flash('message_success', "Insert venue berhasil.");
23  res.redirect('/manajemen_list_venue');
24                              })
25                              .catch(function (err) {
26                                  req.flash('message_err',
27  "Internal server error.");
28  res.redirect('/manajemen_list_venue');
29                              })
30                      }
31                  })
32                  .catch(function (err) {
33                      req.flash('message_err', "Internal
34  server error.");
35                      res.redirect('/manajemen_list_venue');
36                  })
37          }
38          else {
39              req.flash('message_err', "Semua field harus
40  diisi!");
41              res.redirect('/manajemen_list_venue');
42          }
43      }
```

5.2.3 Implementasi Data

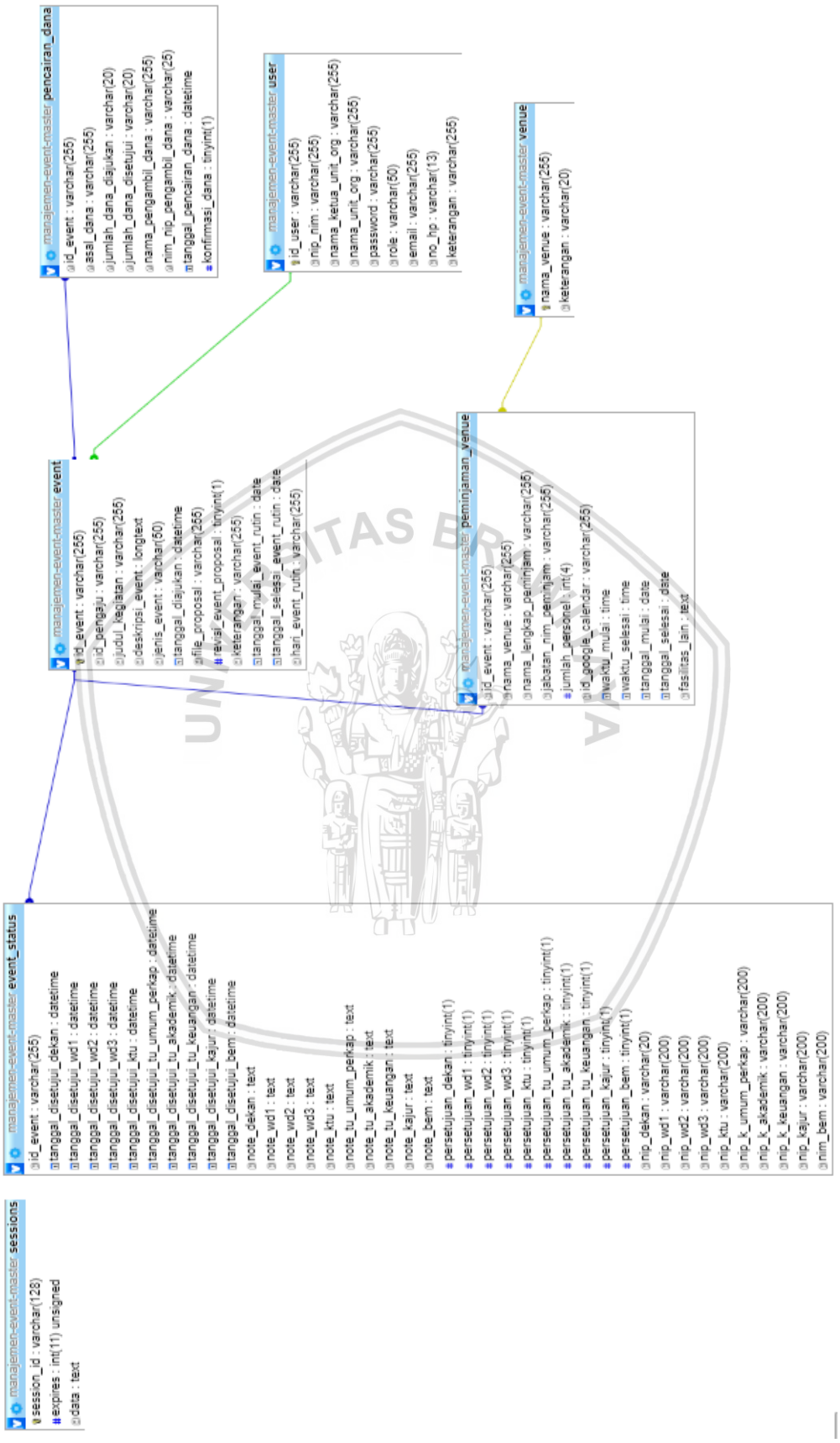
Pada bab ini akan membahas secara detail mengenai *Physical Data Model* (PDM) yang dibuat berdasarkan pada *Entity Relationship Diagram* yang telah dibuat sebelumnya pada bab perancangan. Pada penelitian ini penulis menggunakan database relasional MySQL. Untuk mengimplementasikan skema *database*, penulis menggunakan perangkat lunak bernama XAMPP.

5.2.3.1 *Physical Data Model* (PDM)

Gambar 5.10 dibawah merupakan *Physical Data Model* (PDM) menggambarkan struktur skema *database* sesungguhnya yang digunakan saat implementasi. Pada *entity relationship diagram* di tahap perancangan, diperoleh 5 entitas dan 4 relasi. Kelima entitas yaitu event, venue, user, event_status dan pencairan_dana yang akan dijadikan tabel pada *physical data model*.

Kemudian dari 4 relasi yang teridentifikasi, 1 relasi yaitu relasi "meminjam" yang menghubungkan entitas event dan venue akan dibuat menjadi tabel bernama "peminjaman_venue". Selain itu, penulis akan mengimplementasikan *persistent session* di aplikasi ini. Oleh karena itu, penulis akan membuat tabel bernama *session* yang berisi informasi session dari aktor yang sedang masuk kedalam sistem

Sehingga pada aplikasi ini terdapat 7 tabel yaitu user, sessions, event, event_status, pencairan_dana, pengajuan_event, peminjaman_venue dan venue. Tabel user digunakan untuk menyimpan data akun. Tabel event, event_status, pencairan_dana, peminjaman_venue digunakan untuk menyimpan data pengajuan *event*. Tabel venue digunakan untuk menyimpan seluruh list *venue* yang biasa digunakan untuk menyelenggarakan *event* di FIA UB dan yang terakhir adalah tabel sessions yang digunakan untuk mengimplementasikan *persistent session.*

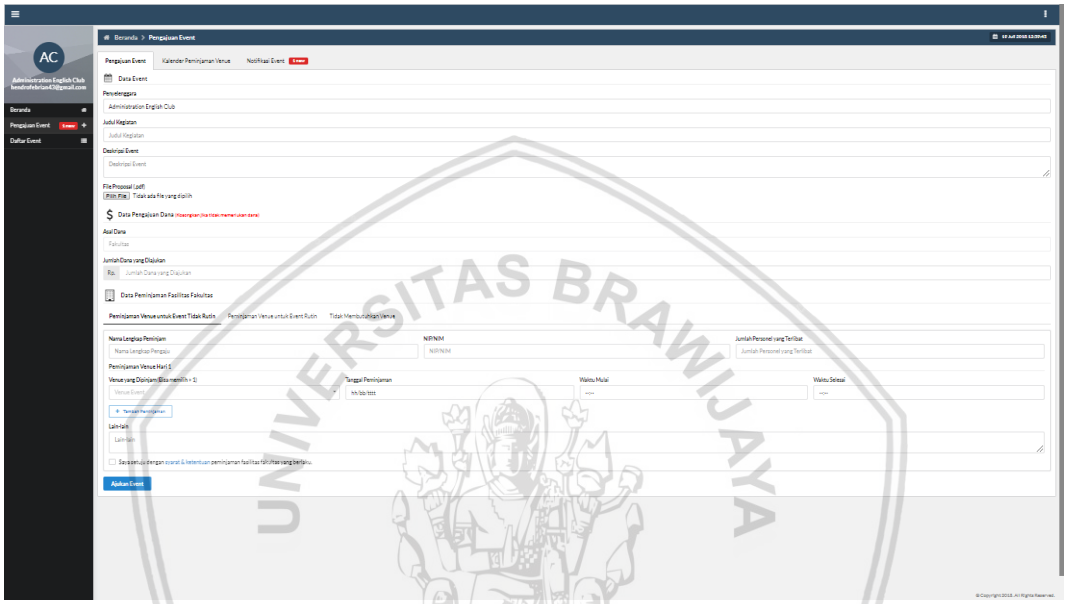


Gambar 5.10 *Physical Data Model*

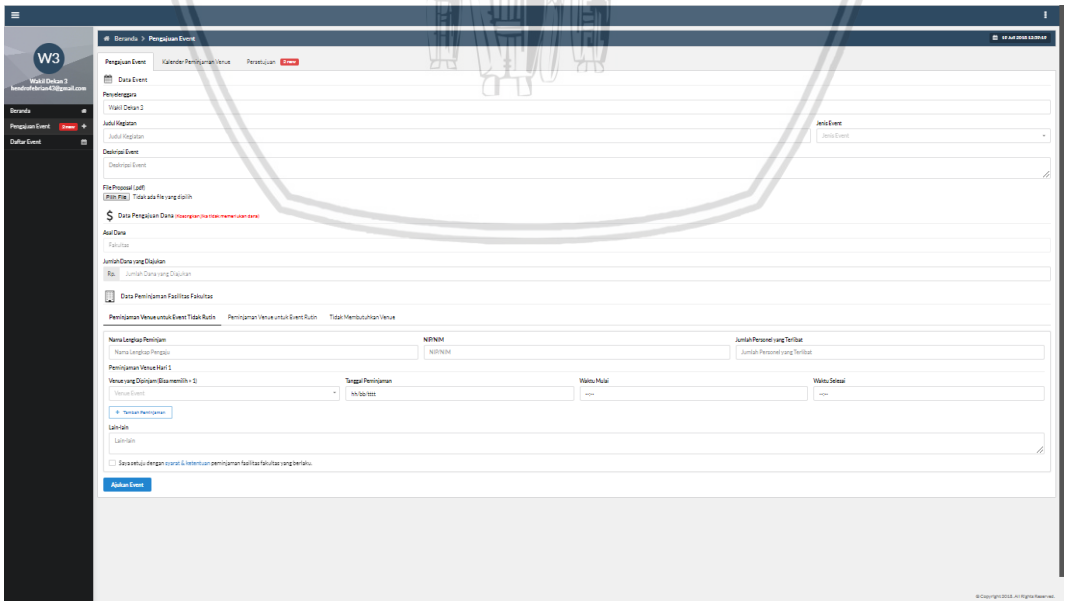
5.2.4 Implementasi Antarmuka

5.2.4.1 Antarmuka Mengajukan *Event*

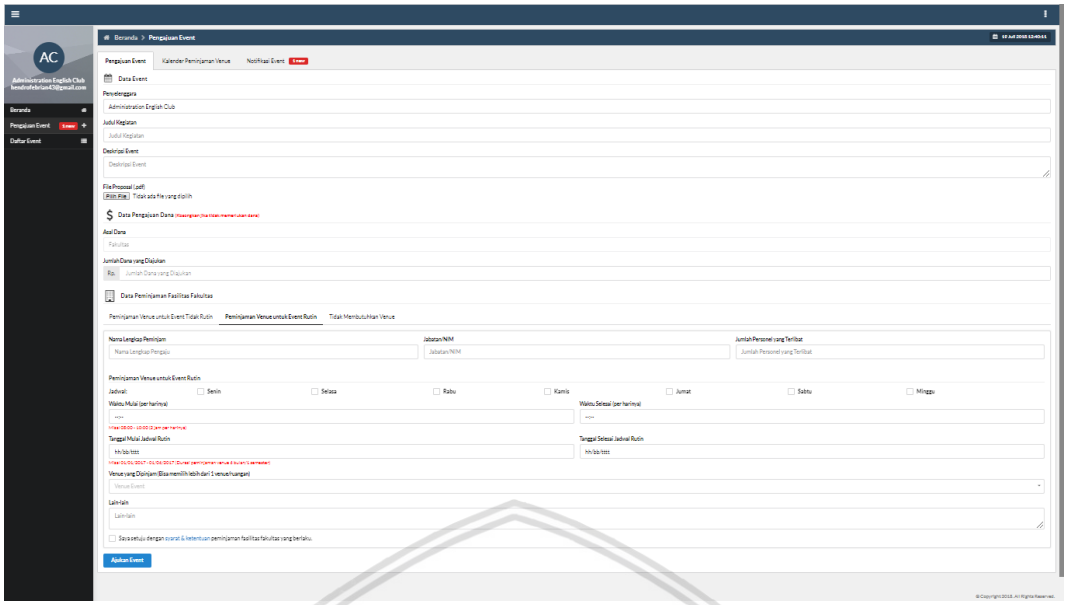
Halaman ini digunakan untuk mengajukan *event* untuk pengaju *event* dari unit internal fakultas (Gambar 5.12) dan pengaju *event* dari organisasi mahasiswa (Gambar 5.11). Aktor dapat memasukkan data peminjaman *venue* pada tab Peminjaman *Venue* untuk *Event* Tidak Rutin (Gambar 5.11 dan Gambar 5.12), Peminjaman *Venue* untuk *Event* Rutin (Gambar 5.13) dan Tidak Membutuhkan *Venue* (Gambar 5.14).



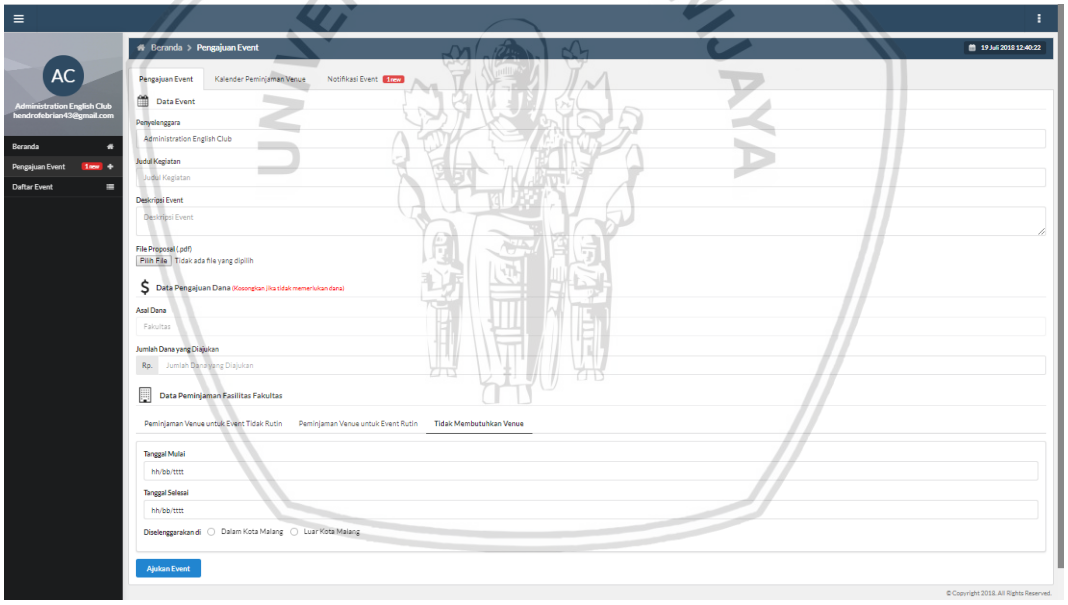
Gambar 5.11 Antarmuka Mengajukan Event Mahasiswa



Gambar 5.12 Antarmuka Mengajukan Event Unit Internal Fakultas



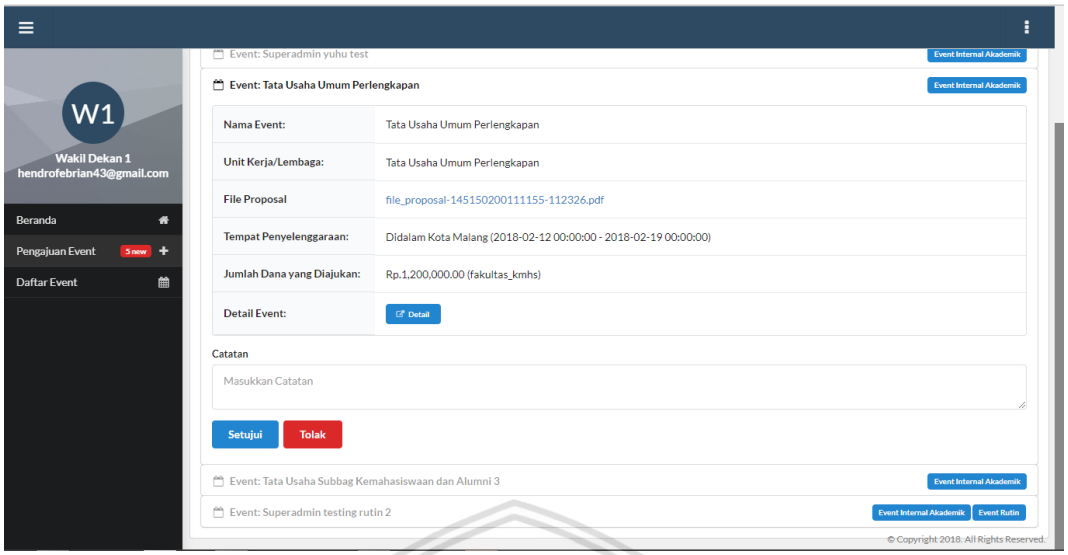
Gambar 5.13 Tab Peminjaman Venue untuk Event Rutin



Gambar 5.14 Tab Tidak Membutuhkan Venue

5.2.4.2 Antarmuka Menyetujui Event

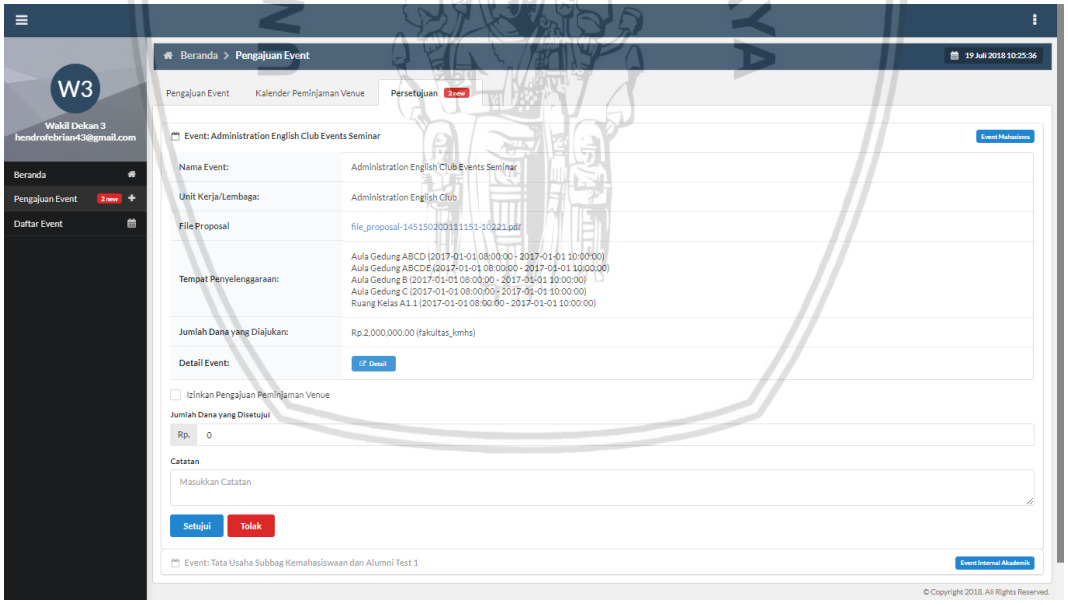
Gambar 5.15 dibawah ini merupakan halaman menyetujui *event* jika dilihat dari akun yang memiliki *role* Dekan, KTU, BEM, Kajur atau Wakil Dekan 1.



Gambar 5.15 Antarmuka Menyetujui Event

5.2.4.3 Antarmuka Menyetujui Event dari WD2 dan WD3

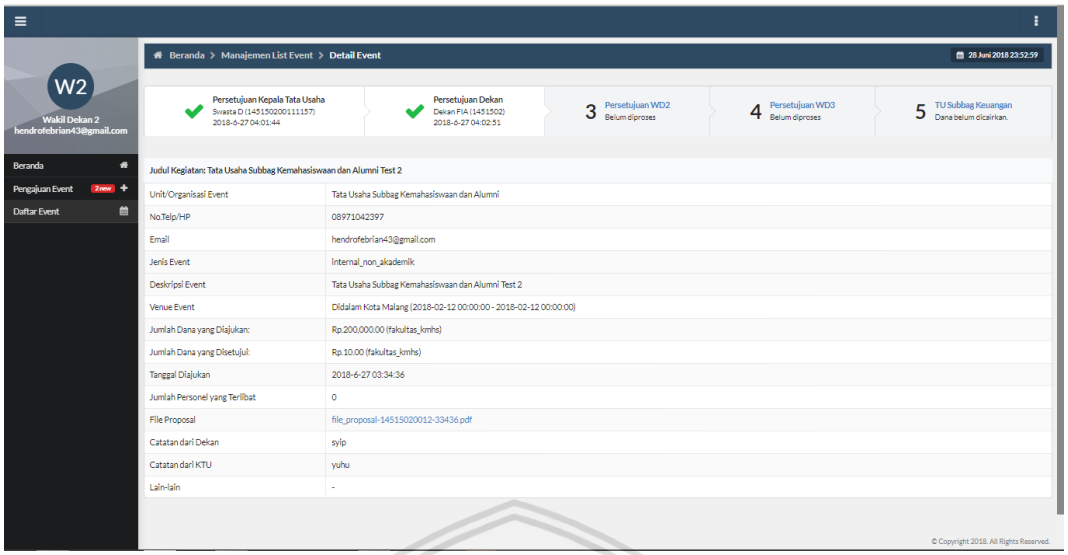
Gambar 5.16 dibawah ini merupakan halaman menyetujui *event* jika dilihat dari akun yang memiliki *role* WD2 atau WD3.



Gambar 5.16 Antarmuka Menyetujui Event dari WD2 dan WD3

5.2.4.4 Antarmuka Melihat Detail Event

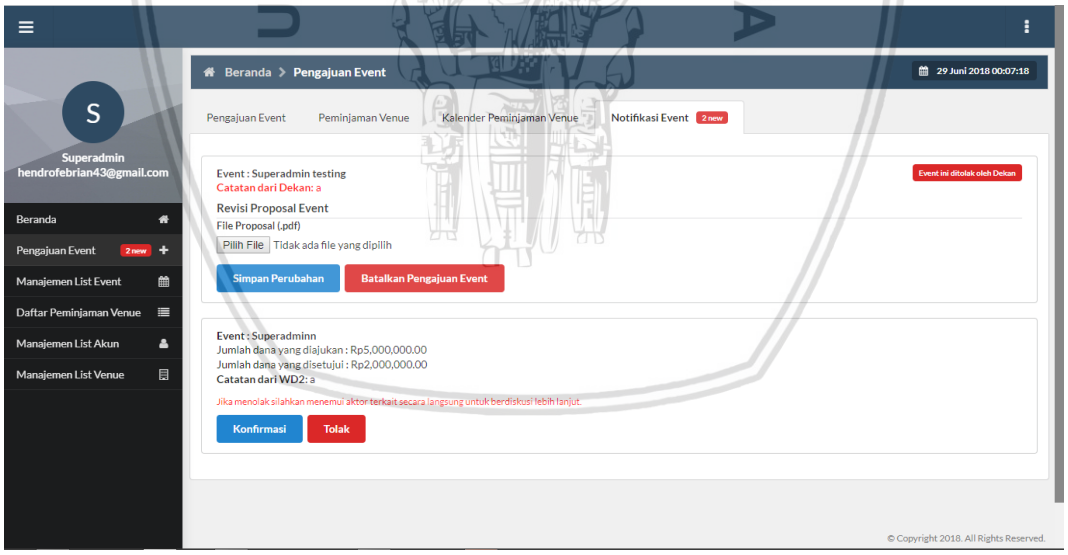
Gambar 5.17 dibawah ini merupakan halaman detail *event*. Pada halaman ini terdapat seluruh data *event*, siapa saja aktor yang harus menyetujui *event* dan status dari *event* tersebut.



Gambar 5.17 Antarmuka Melihat Detail Event

5.2.4.5 Antarmuka Melihat Halaman Notifikasi

Gambar 5.18 dibawah ini adalah Halaman yang digunakan pengaju *event* untuk merevisi proposal atau menyetujui serta menolak dana yang disetujui oleh *Approver* Dana (WD2, WD3).



Gambar 5.18 Antarmuka Melihat Halaman Notifikasi

BAB 6 PENGUJIAN

Pengujian merupakan tahapan yang bertujuan untuk memeriksa apakah seluruh kebutuhan fungsional dan non-fungsional yang telah didefinisi dan dispesifikasikan sudah berjalan sesuai harapan. Penulis akan menggunakan metode *manual testing* (tidak menggunakan *tools*) untuk menguji kebutuhan fungsional pada penelitian ini. Sedangkan untuk kebutuhan non-fungsional akan diuji secara *automate* (menggunakan *tools*). Fase pengujian pada penelitian ini akan dibagi menjadi 3 tahap yaitu pengujian *unit*, pengujian sistem/validasi dan pengujian *browser compatibility*.

6.1 Pengujian Unit

Pada pengujian *unit* di penelitian ini, penulis menggunakan metode *basis path testing*. Untuk mengidentifikasi independent path yang ada, akan dilakukan dengan membuat *flow graph*. *Flow graph* dibuat berdasarkan *pseudocode* yang sebelumnya telah dibuat pada tahap perancangan.

6.1.1 Pengujian Unit Method Melihat Halaman Persetujuan

6.1.1.1 Pseudocode

Nama Kelas: Approver_Controller

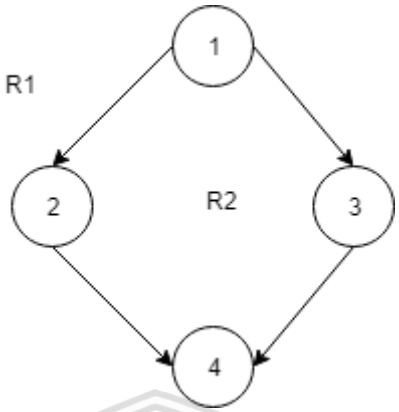
Nama Method: `melihatHalamanPersetujuan()`

Tabel 6.1 *Pseudocode Method* Melihat Halaman Persetujuan

No	*Pseudocode Method* Melihat Halaman Persetujuan
1	`Deklarasi var pengajuan_event`
2	`Deklarasi var params`
3	`Deklarasi var total`
4	`Pengajuan_event= new Pengajuan_Event(params)`
5	`Panggil method getDataHalamanPersetujuan()`
6	`If (result > 0)`
7	`Session notif <- result.length`
8	`Render halaman persetujuan`
9	`Else`
10	`Generate notifikasi ”Belum ada event”`
11	`Render halaman persetujuan`
12	`Endif`

6.1.1.2 Basis Path Testing

a. Flow Graph



Gambar 6.1 *Flow Graph* Melihat Halaman Persetujuan

b. Cyclomatic Complexity (V(G))

1. V(G) = jumlah *region* = 2
2. V(G) = jumlah *edge* - jumlah *node* + 2 = 4 – 4 + 2 = 2
3. V(G) = jumlah *predicate node* + 1 = 1 + 1 = 2

c. Independent Path

1. 1-2-4
2. 1-3-4

Tabel 6.2 Hasil Pengujian *Unit Method* Melihat Halaman Persetujuan

No.	No. Jalur	Data Pengujian	*Expected Result*	*Result*	Status
1	1	Result > 0	Sistem memanggil *method* `getDataHalaman Persetujuan()`, sistem menginisialisasi nilai dari *session* notif dan sistem me*render* halaman persetujuan	Sistem memanggil *method* untuk mengambil data persetujuan di *database,* sistem menginisialisasi nilai dari *session* notif dan sistem me*render* halaman persetujuan	*Valid*
2	2	Result < 0	Sistem me*render* halaman persetujuan dan menampilkan notifikasi berisi pesan “Belum ada	Sistem me*render* halaman persetujuan dan menampilkan notifikasi berisi pesan “Belum ada	*Valid*

			event"	*event*"	

6.1.2 Pengujian Unit Method Mengkonfirmasi Dana

6.1.2.1 Pseudocode

Nama Kelas: Pengaju_Event_Controller

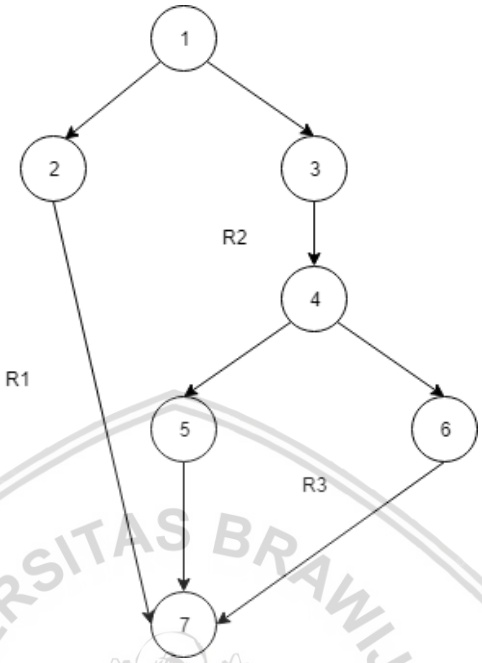
Nama Method: `mengkonfirmasiDana()`

Tabel 6.3 *Pseudocode Method* Mengkonfirmasi Dana

No	***Pseudocode Method*** **Mengkonfirmasi Dana**
1	`Deklarasi var pengajuan_event`
2	`Params <- {id_event : ID event yang ada di URL}`
3	`Deklarasi var total`
4	`Pengajuan event = new Pengajuan_Event_Controller(params)`
5	`If(button konfirmasi dana diklik)`
6	`Panggil method mengkonfirmasiDana menggunakan objek`
7	`pengajuan_event`
8	`Decrement session yang menyimpan jumlah notifikasi`
9	`Generate pesan notifikasi berhasil`
10	`Arahkan aktor ke halaman pengajuan event`
11	`Elif(button tolak dana diklik)`
12	`Panggil method menolakKonfirmasiDana menggunakan`
13	`objek pengajuan_event`
14	`Panggil method getDataPencairanDana menggunakan objek`
15	`pengajuan_event`
16	`if(asal dana adalah fakultas/kemahasiswaan dan pengaju`
17	`event adalah organisasi mahasiswa)`
18	`panggil method resetpersetujuanWD3`
19	`elif(asal dana adalah fakultas/kemahasiswaan dan`
20	`pengaju event adalah unit internal)`
21	`panggil method resetPersetujuanWD2`
22	`Endif`
23	`Decrement session yang menyimpan jumlah notifikasi`
24	`Generate pesan berhasil`
25	`Arahkan aktor ke halaman pengajuan event`
26	`Endif`

6.1.2.2 Basis Path Testing

a. Flow Graph



Gambar 6.2 *Flow Graph* Mengkonfirmasi Dana

b. Cyclomatic Complexity (V(G))

1. V(G) = jumlah *region* = 3
2. V(G) = jumlah *edge* - jumlah *node* + 2 = 8 – 7 + 2 = 3
3. V(G) = jumlah *predicate node* + 1 = 2 + 1 = 3

c. Independent Path

1. 1-2-7
2. 1-3-4-5-7
3. 1-3-4-6-7

Tabel 6.4 Hasil Pengujian *Unit Method* Mengkonfirmasi Dana

No.	No. Jalur	Data Pengujian	*Expected Result*	*Result*	Status
1	1	Konfirmasi dana != undefined && konfirmasi dana !== null	Sistem memanggil *method* `mengkonfirmasiDana()`, mengarahkan aktor ke halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil	Sistem memanggil *method* `mengkonfirmasiDana()`, mengarahkan aktor ke halaman pengajuan event	*Valid*

			menyetujui dana".	dan menampilkan notifikasi berisi pesan "Berhasil menyetujui dana".	
2	2	Tolak dana != undefined && tolak dana != null && (role == "pengajuevent_mhs" || role == "bem_fia") && asal_dana == "fakultas_kmhs"	Sistem memanggil *method* `mengkonfirmasiDana()`, `getDataPencairanDana()`, `resetPersetujuanWD3`, mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menolak dana".	Sistem memanggil *method* `mengkonfirmasiDana()`, `getDataPencairanDana()`, `resetPersetujuanWD3`, mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menolak dana".	*Valid*
3	3	Tolak dana != undefined && tolak dana != null && (role != "pengajuevent_mhs" && role != "bem_fia") && asal_dana == "fakultas_kmhs"	Sistem memanggil *method* `mengkonfirmasiDana()`, `getDataPencairanDana()`, `resetPersetujuanWD2`, mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menolak dana".	Sistem memanggil *method* `mengkonfirmasiDana()`, `getDataPencairanDana()`, `resetPersetujuanWD2`, mengarahkan aktor ke halaman pengajuan *event* dan menampilkan notifikasi berisi pesan "Berhasil menolak dana".	*Valid*

6.1.3 Pengujian *Unit Method* Memasukkan *Venue* Baru

6.1.3.1 Pseudocode

Nama Kelas: SuperAdmin_Controller

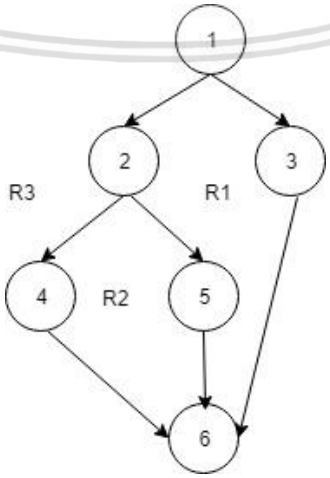
Nama Method: `mengkonfirmasiDana()`

Tabel 6.5 Pseudocode *Method* Memasukkan *Venue* Baru

No	*Pseudocode Method* Memasukkan *Venue* Baru
1	`Nama venue <- nama venue yang diinputkan user`
2	`Keterangan <- keterangan yang diinputkan user`
3	`Params <- {nama venue: nama venue,`
4	`Keterangan : keterangan}`
5	`venue = new Venue_Model(params)`
6	`If(exist(nama_venue)&& exist(keterangan)`
7	`Panggil method getVenueByNama`
8	`If(exist(result))`
9	`Generate pesan error venue telah ada di db`
10	`Arahkan aktor ke halaman List Venue`
11	`Else`
12	`Panggil method insertNewVenue`
13	`Generate pesan sukses`
14	`Arahkan aktor ke halaman List Venue`
15	`Endif`
16	`Else`
17	`Generate pesan error`
18	`Arahkan aktor ke halaman List Venue`
19	`Endif`

6.1.3.2 Basis Path Testing

a. Flow Graph



Gambar 6.3 *Flow Graph* Memasukkan *Venue* Baru

b. *Cyclomatic Complexity* (V(G))

1. V(G) = jumlah *region* = 3
2. V(G) = jumlah *edge* - jumlah *node* + 2 = 7 – 6 + 2 = 3
3. V(G) = jumlah *predicate node* + 1 = 2 + 1 = 3

c. *Independent Path*

1. 1-3-6
2. 1-2-4-6
3. 1-2-5-6

Tabel 6.6 Hasil Pengujian *Unit Method* Memasukkan *Venue* Baru

No.	No. Jalur	Data Pengujian	*Expected Result*	*Result*	Status
1	1	Nama_venue == result. nama_venue	Sistem memanggil *method* `getVenueByNama()`, mengarahkan aktor ke halaman *list venue* dan menampilkan notifikasi berisi pesan "Nama *venue* telah ada di *database*".	Sistem memanggil *method* `getVenueByNama()`, mengarahkan aktor ke halaman *list venue* dan menampilkan notifikasi berisi pesan "Nama *venue* telah ada di *database*".	*Valid*
2	2	Nama_venue != result. nama_venue	Sistem memanggil *method* `getVenueByNama()`, sistem memanggil *method* `insertNewVenue()`, sistem mengarahkan aktor ke halaman *list venue* dan menampilkan notifikasi berisi pesan "Berhasil memasukkan *venue*".	Sistem memanggil *method* `getVenueByNama()`, sistem memanggil *method* `insertNewVenue()`, sistem mengarahkan aktor ke halaman *list venue* dan menampilkan notifikasi berisi pesan "Berhasil memasukkan *venue*".	*Valid*
3	3	Nama_venue == null atau Keterangan	Sistem mengarahkan aktor ke halaman *list*	Sistem mengarahkan aktor ke halaman *list venue* dan	*Valid*

		== null	*venue* dan menampilkan notifikasi berisi pesan "Tidak boleh ada *field* yang dikosongkan".	menampilkan notifikasi berisi pesan "Tidak boleh ada *field* yang dikosongkan".	

6.2 Pengujian Sistem/Validasi

Pada pengujian validasi atau pengujian sistem, penulis akan menggunakan metode *functional testing.* Penulis akan menguji setiap definisi kebutuhan yang ada serta skenario alternatif dari masing-masing definisi kebutuhan yang terdapat di *use case scenario*. Berikut ini merupakan hasil dari pengujian sistem/validasi pada penelitian ini. Pada penelitian ini penulis akan menguji 59 kebutuhan fungsional beserta seluruh *alternative flow* yang ada pada setiap kebutuhan yang terdapat di *use case scenario*.

Tabel 6.7 Pengujian Validasi Login

Kode Kebutuhan	AME_F_100
Nama Kasus Uji	*Login*
Prosedur	1. Aktor mengakses halaman *login.* 2. Aktor mengisi *form login* yang terdiri dari *field* NIP/NIM dan *password* kemudian mengklik *button login*.
Expected Result	Aktor terautentikasi dan terautorisasi, sistem menampilkan halaman beranda dan menampilkan pesan "Welcome, username!".
Result	Aktor terautentikasi dan terautorisasi, sistem menampilkan halaman beranda dan menampilkan pesan "Welcome, username!".
Status	*Valid*

Tabel 6.8 Pengujian Validasi *Login* Alternatif 1

Kode Kebutuhan	AME_F_100
Nama Kasus Uji	Login Alternatif 1
Prosedur	1. Aktor mengakses halaman login. 2. Aktor mengisi field NIP/NIM atau *password* dengan nilai yang tidak *valid* (misalnya ada *field* yang dikosongkan) kemudian menekan tombol *login*.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error.*

Result	Sistem menampilkan notifikasi berisi pesan *error.*
Status	*Valid*

Tabel 6.9 Pengujian Validasi *Login* Alternatif 2

Kode Kebutuhan	AME_F_100
Nama Kasus Uji	Login Alternatif 2
Prosedur	1. Aktor mengakses halaman login. 2. Aktor mengisi *field* NIP/NIM yang salah (tidak terdaftar di sistem) dan mengisi *password.*
Expected Result	Sistem menampilkan halaman *login* dan notifikasi berisi pesan *error* "NIP/NIM salah".
Result	Sistem menampilkan halaman *login* dan notifikasi berisi pesan *error* "NIP/NIM salah".
Status	*Valid*

Tabel 6.10 Pengujian Validasi *Login* Alternatif 3

Kode Kebutuhan	AME_F_100
Nama Kasus Uji	Login Alternatif 3
Prosedur	1. Aktor mengakses halaman login. 2. Aktor mengisi field NIP/NIM yang terdaftar dan *password* yang salah (tidak sesuai dengan *password* akun di *database*).
Expected Result	Sistem menampilkan halaman login dan notifikasi berisi pesan *error "*Password salah*".*
Result	Sistem menampilkan halaman login dan notifikasi berisi pesan *error "*Password salah*".*
Status	*Valid*

Tabel 6.11 Pengujian Validasi Logout

Kode Kebutuhan	AME_F_200
Nama Kasus Uji	Logout
Prosedur	1. Aktor mengklik tombol logout
Expected Result	Sistem menghapus data session aktor dari database kemudian sistem menampilkan halaman beranda

Result	Sistem menghapus data session aktor dari database kemudian sistem menampilkan halaman beranda
Status	*Valid*

Tabel 6.12 Pengujian Validasi Melihat Beranda

Kode Kebutuhan	AME_F_300
Nama Kasus Uji	Melihat Beranda
Prosedur	1. Aktor mengakses halaman beranda
Expected Result	Sistem menampilkan halaman beranda
Result	Sistem menampilkan halaman beranda
Status	*Valid*

Tabel 6.13 Pengujian Validasi Melihat *List Event* dari Pengaju *Event*

Kode Kebutuhan	AME_F_400
Nama Kasus Uji	Melihat *List Event* dari Pengaju *Event*
Prosedur	1. Aktor mengakses halaman *list event*
Expected Result	Sistem menampilkan list *event* yang pernah diajukan aktor
Result	Sistem menampilkan list *event* yang pernah diajukan aktor
Status	*Valid*

Tabel 6.14 Pengujian Validasi Melihat *List Event* dari Pengaju *Event* Alternatif 1

Kode Kebutuhan	AME_F_400
Nama Kasus Uji	Melihat *List Event* dari Pengaju *Event* Alternatif 1
Prosedur	1. Aktor yang belum pernah mengajukan *event* mengakses halaman *list event*
Expected Result	Sistem menampilkan halaman *list event* dan menampilkan notifikasi berisi pesan "*No data available in table*"
Result	Sistem menampilkan halaman *list event* dan menampilkan notifikasi berisi pesan "*No data available in table*"
Status	*Valid*

Tabel 6.15 Pengujian Validasi Melihat Halaman Persetujuan Peminjaman *Venue*

Kode Kebutuhan	AME_F_500

Nama Kasus Uji	Melihat Halaman Persetujuan Peminjaman *Venue*
Prosedur	1. Aktor mengakses halaman persetujuan peminjaman *venue*
Expected Result	Sistem menampilkan *list event* yang memerlukan persetujuan peminjaman *venue* dari aktor
Result	Sistem menampilkan *list event* yang memerlukan persetujuan peminjaman *venue* dari aktor
Status	*Valid*

Tabel 6.16 Pengujian Validasi Melihat Halaman Persetujuan Peminjaman *Venue* Alternatif 1

Kode Kebutuhan	AME_F_500
Nama Kasus Uji	Melihat Halaman Persetujuan Peminjaman Venue Alternatif 1
Prosedur	1. Tidak ada peminjaman venue yang memerlukan persetujuan dan aktor mengakses halaman persetujuan peminjaman venue
Expected Result	Sistem menampilkan halaman peminjaman *venue* dan menampilkan notifikasi "Belum ada *event*"
Result	Sistem menampilkan halaman peminjaman *venue* dan menampilka notifikasi "Belum ada *event*"
Status	*Valid*

Tabel 6.17 Pengujian Validasi Melihat Halaman Persetujuan Event

Kode Kebutuhan	AME_F_600
Nama Kasus Uji	Melihat Halaman Persetujuan Event
Prosedur	1. Aktor mengakses halaman persetujuan *event*
Expected Result	Sistem menampilkan *list event* yang memerlukan persetujuan aktor
Result	Sistem menampilkan *list event* yang memerlukan persetujuan aktor
Status	*Valid*

Tabel 6.18 Pengujian Validasi Melihat Halaman Persetujuan *Event* Alternatif 1

Kode Kebutuhan	AME_F_600

Nama Kasus Uji	Melihat Halaman Persetujuan *Event* Alternatif 1
Prosedur	1. Tidak ada *event* yang memerlukan persetujuan dan aktor mengakses halaman persetujuan *event*
Expected Result	Sistem menampilkan halaman persetujuan *event* dan menampilkan notifikasi "Belum ada *event*"
Result	Sistem menampilkan halaman persetujuan *event* dan menampilkan notifikasi "Belum ada *event*"
Status	*Valid*

Tabel 6.19 Pengujian Validasi Memfilter Event di List Event

Kode Kebutuhan	AME_F_700
Nama Kasus Uji	Memfilter Event di List Event
Prosedur	1. Aktor mengakses halaman list event 2. Aktor memilih kategori "Tampilkan hanya event saya"
Expected Result	Sistem hanya menampilkan seluruh event yang pernah diajukan aktor tersebut
Result	Sistem henya menampilkan seluruh event yang pernah diajukan aktor tersebut
Status	*Valid*

Tabel 6.20 Pengujian Validasi Memfilter Event di List Event Alternatif 1

Kode Kebutuhan	AME_F_700
Nama Kasus Uji	Memfilter Event di List Event Alternatif 1
Prosedur	1. Aktor mengakses halaman list event. 2. Aktor belum pernah mengajukan event 3. Aktor memilih kategori "Tampilkan hanya event saya"
Expected Result	Sistem menampilkan pesan "*No data available in table*"
Result	Sistem menampilkan pesan "*No data available in table*"
Status	*Valid*

Tabel 6.21 Pengujian Validasi Memfilter Event di List Event Alternatif 2

Kode Kebutuhan	AME_F_700
Nama Kasus Uji	Memfilter Event di List Event Alternatif 2
Prosedur	1. Login menggunakan akun Dekan, WD1, WD2 atau

	WD3 2. Aktor mengakses halaman list event. 3. Aktor memilih kategori “Tampilkan seluruh event”
Expected Result	Sistem menampilkan seluruh list event yang ada di sistem
Result	Sistem menampilkan seluruh list event yang ada di sistem
Status	*Valid*

Tabel 6.22 Pengujian Validasi Memfilter Event di List Event Alternatif 3

Kode Kebutuhan	AME_F_700
Nama Kasus Uji	Memfilter Event di List Event Alternatif 3
Prosedur	1. Login menggunakan akun TU Subbag Kemahasiswaan, Kajur, KTU, BEM, TU Subbag Akademik, TU Subbag Umum dan Perlengkapan, TU Subbag Keuangan 2. Aktor mengakses halaman list event. 3. Aktor memilih kategori “Tampilkan seluruh event”
Expected Result	Sistem menampilkan seluruh list event yang pernah diajukan aktor dan list event yang pernah disetujui oleh aktor tersebut
Result	Sistem menampilkan seluruh list event yang pernah diajukan aktor dan list event yang pernah disetujui oleh aktor tersebut
Status	*Valid*

Tabel 6.23 Pengujian Validasi Melihat Halaman Persetujuan Event dari WD2 dan WD3

Kode Kebutuhan	AME_F_800
Nama Kasus Uji	Melihat Halaman Persetujuan Event dari WD2 dan WD3
Prosedur	1. Aktor mengakses halaman persetujuan event
Expected Result	Sistem menampilkan list event yang memerlukan persetujuan WD2 atau WD3
Result	Sistem menampilkan list event yang memerlukan persetujuan WD2 atau WD3
Status	Valid

Tabel 6.24 Pengujian Validasi Melihat Halaman Persetujuan Event dari WD2 dan WD3 Alternatif 1

Kode Kebutuhan	AME_F_800

Nama Kasus Uji	Melihat Halaman Persetujuan Event dari WD2 dan WD3 Alternatif 1
Prosedur	1. Tidak ada event yang memerlukan persetujuan dan aktor mengakses halaman persetujuan event
Expected Result	Sistem menampilkan halaman peminjaman venue dan menampilka notifikasi "Belum ada event"
Result	Sistem menampilkan halaman peminjaman venue dan menampilka notifikasi "Belum ada event"
Status	Valid

Tabel 6.25 Pengujian Validasi Melihat Detail *Event*

Kode Kebutuhan	AME_F_900
Nama Kasus Uji	Melihat Detail *Event*
Prosedur	1. Aktor mengklik *button* detail *event*
Expected Result	Sistem menampilkan detail dari event yang dipilih
Result	Sistem menampilkan detail dari event yang dipilih
Status	*Valid*

Tabel 6.26 Pengujian Validasi Mencari Event

Kode Kebutuhan	AME_F_1000
Nama Kasus Uji	Mencari *Event*
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor mengisi *field* pencarian *event*
Expected Result	Sistem menampilkan *event* yang dicari aktor
Result	Sistem menampilkan *event* yang dicari aktor
Status	*Valid*

Tabel 6.27 Pengujian Validasi Mencari Event Alternatif 1

Kode Kebutuhan	AME_F_1000
Nama Kasus Uji	Mencari *Event* Alternatif 1
Prosedur	1. Aktor mencari event yang tidak ada di sistem
Expected Result	Sistem menampilkan notifikasi "*No matching records found*"
Result	Sistem menampilkan notifikasi "*No matching records found*"

Status	*Valid*

Tabel 6.28 Pengujian Validasi Mengajukan Event

Kode Kebutuhan	AME_F_1100
Nama Kasus Uji	Mengajukan Event
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, nama lengkap peminjam, NIP/NIM peminjam, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol ajukan *event*.
Expected Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"
Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"
Status	*Valid*

Tabel 6.29 Pengujian Validasi Mengajukan Event Alternatif 1

Kode Kebutuhan	AME_F_1100
Nama Kasus Uji	Mengajukan Event Alternatif 1
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi field di form pengajuan event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.30 Pengujian Validasi Mengajukan Event Alternatif 2

Kode Kebutuhan	AME_F_1100

Nama Kasus Uji	Mengajukan Event Alternatif 2
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, nama lengkap peminjam, NIP/NIM peminjam, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, hari, lain-lain dan checkbox S&K serta menekan tombol ajukan *event*.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.31 Pengujian Validasi Mengajukan Event Alternatif 3

Kode Kebutuhan	AME_F_1100
Nama Kasus Uji	Mengajukan Event Alternatif 3
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, nama lengkap peminjam, NIP/NIM peminjam, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol ajukan *event*. 3. Data peminjaman venue yang diinputkan aktor berbenturan dengan peminjaman venue event lain
Expected Result	Sistem tidak menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Venue yang anda pilih akan dipakai oleh event nama event pada tanggal mulai – tanggal selesai”
Result	Sistem tidak menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Venue yang anda pilih akan dipakai oleh event nama event pada tanggal mulai – tanggal selesai”
Status	*Valid*

Tabel 6.32 Pengujian Validasi Mengajukan Event Alternatif 4

Kode Kebutuhan	AME_F_1100
Nama Kasus Uji	Mengajukan Event Alternatif 4
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, nama lengkap peminjam, NIP/NIM peminjam, jumlah personel yang terlibat, hari, *venue* yang dipinjam, tanggal mulai event rutin, tanggal selesai event rutin, waktu mulai event rutin, waktu selesai event rutin, lain-lain dan checkbox S&K serta menekan tombol ajukan *event*.
Expected Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"
Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"
Status	*Valid*

Tabel 6.33 Pengujian Validasi Mengajukan Event Alternatif 5

Kode Kebutuhan	AME_F_1100
Nama Kasus Uji	Mengajukan Event Alternatif 5
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, file proposal, asal dana, jumlah dana yang diajukan, tempat diselenggarakan, tanggal mulai dan tanggal selesai serta menekan tombol ajukan *event*.
Expected Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"
Result	Sistem berhasil menyimpan data pengajuan event ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil mengajukan event"

Status	*Valid*

Tabel 6.34 Pengujian Validasi Mengedit Data Profil

Kode Kebutuhan	AME_F_1200
Nama Kasus Uji	Mengedit Data Profil
Prosedur	1. Aktor mengakses halaman edit data profil 2. Aktor mengubah field NIP/NIM dengan NIP/NIM yang belum ada di database kemudian menekan tombol simpan perubahan
Expected Result	Sistem berhasil menyimpan perubahan data profil yang dilakukan aktor kemudian sistem menampilkan halaman edit profil dan notifikasi berisi pesan "Berhasil mengedit data profil"
Result	Sistem berhasil menyimpan perubahan data profil yang dilakukan aktor kemudian sistem menampilkan halaman edit profil dan notifikasi berisi pesan "Berhasil mengedit data profil"
Status	*Valid*

Tabel 6.35 Pengujian Validasi Mengedit Data Profil Alternatif 1

Kode Kebutuhan	AME_F_1200
Nama Kasus Uji	Mengedit Data Profil Alternatif 1
Prosedur	1. Aktor mengakses halaman edit data profil 2. Aktor mengubah field NIP/NIM dengan NIP/NIM yang sudah ada di database kemudian menekan tombol simpan perubahan
Expected Result	Sistem tidak menyimpan perubahan ke database kemudian sistem menampilkan halaman edit profil dan menampilkan notifikasi berisi pesan "NIP/NIM telah digunakan"
Result	Sistem tidak menyimpan perubahan ke database kemudian sistem menampilkan halaman edit profil dan menampilkan notifikasi berisi pesan "NIP/NIM telah digunakan"
Status	*Valid*

Tabel 6.36 Pengujian Validasi Mengedit Data Profil Alternatif 2

Kode Kebutuhan	AME_F_1200

Nama Kasus Uji	Mengedit Data Profil Alternatif 2
Prosedur	1. Aktor mengakses halaman edit data profil 2. Aktor mengisi *field* yang ada di form edit data profil dengan nilai yang tidak *valid* (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*.
Result	Sistem menampilkan notifikasi berisi pesan *error*.
Status	*Valid*

Tabel 6.37 Pengujian Validasi Mengedit Password

Kode Kebutuhan	AME_F_1300
Nama Kasus Uji	Mengedit Password
Prosedur	1. Aktor mengakses halaman edit data event 2. Aktor mengklik tab edit password 3. Aktor mengisi field password lama, password baru dan konfirmasi password baru
Expected Result	Sistem berhasil penyimpan perubahan password akun ke database kemudian sistem menampilkan halaman edit data profil dan menampilkan notifikasi berisi pesan "Berhasil mengubah *password*"
Result	Sistem berhasil penyimpan perubahan password akun ke database kemudian sistem menampilkan halaman edit data profil dan menampilkan notifikasi berisi pesan "Berhasil mengubah *password*"
Status	*Valid*

Tabel 6.38 Pengujian Validasi Mengedit Password Alternatif 1

Kode Kebutuhan	AME_F_1300
Nama Kasus Uji	Mengedit Password Alternatif 1
Prosedur	1. Aktor mengakses halaman edit *data event* 2. Aktor mengklik tab edit password 3. Aktor mengisi *field* di form edit password dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error

Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.39 Pengujian Validasi Mengedit Password Alternatif 2

Kode Kebutuhan	AME_F_1300
Nama Kasus Uji	Mengedit Password Alternatif 2
Prosedur	1. Aktor mengakses halaman edit data event 2. Aktor mengklik tab edit password 3. Aktor tidak mengisi field password lama yang tidak sesuai dengan password akun di database kemudian menekan tombol simpan perubahan
Expected Result	Sistem menampilkan notifikasi berisi pesan error "Password lama salah"
Result	Sistem menampilkan notifikasi berisi pesan error "Password lama salah"
Status	*Valid*

Tabel 6.40 Pengujian Validasi Mengedit Password Alternatif 3

Kode Kebutuhan	AME_F_1300
Nama Kasus Uji	Mengedit Password Alternatif 3
Prosedur	1. Aktor mengakses halaman edit data event 2. Aktor mengklik tab edit password 3. Aktor tidak mengisi field password baru dan konfirmasi password baru dengan nilai yang sama
Expected Result	Sistem menampilkan notifikasi berisi pesan error "*Password* baru tidak sama dengan konfirmasi *password*"
Result	Sistem menampilkan notifikasi berisi pesan error "*Password* baru tidak sama dengan konfirmasi *password*"
Status	*Valid*

Tabel 6.41 Pengujian Validasi Melihat Jumlah Notifikasi

Kode Kebutuhan	AME_F_1400
Nama Kasus Uji	Melihat Jumlah Notifikasi
Prosedur	1. Aktor mengakses salah satu halaman di sistem misalnya halaman beranda

Expected Result	Sistem menampilkan jumlah notifikasi yang berada di halaman notifikasi
Result	Sistem menampilkan jumlah notifikasi yang berada di halaman notifikasi
Status	*Valid*

Tabel 6.42 Pengujian Validasi Mengedit Peminjaman Venue

Kode Kebutuhan	AME_F_1500
Nama Kasus Uji	Mengedit Peminjaman Venue
Prosedur	1. Aktor mengakses halaman edit peminjaman venue 2. Aktor mengisi *field* nama venue, tanggal peminjaman, waktu mulai atau waktu selesai dan menekan tombol simpan perubahan
Expected Result	Sistem menyimpan perubahan data peminjaman venue ke database dan menampilkan halaman list peminjaman venue serta menampilkan notifikasi berisi pesan "Berhasil mengedit peminjaman venue"
Result	Sistem menyimpan perubahan data peminjaman venue ke database dan menampilkan halaman list peminjaman venue serta menampilkan notifikasi berisi pesan "Berhasil mengedit peminjaman venue"
Status	*Valid*

Tabel 6.43 Pengujian Validasi Mengedit Peminjaman Venue Alternatif 1

Kode Kebutuhan	AME_F_1500
Nama Kasus Uji	Mengedit Peminjaman Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman edit peminjaman venue dari peminjaman venue telah dimasukkan kedalam Google Calendar 2. Aktor mengisi *field* nama venue, tanggal peminjaman, waktu mulai atau waktu selesai dan menekan tombol simpan perubahan
Expected Result	Sistem menyimpan perubahan data peminjaman venue ke database, menyimpan perubahan di Google Calendar dan menampilkan halaman list peminjaman venue serta menampilkan notifikasi berisi pesan "Berhasil mengedit peminjaman venue"

Result	Sistem menyimpan perubahan data peminjaman venue ke database, menyimpan perubahan di Google Calendar dan menampilkan halaman list peminjaman venue serta menampilkan notifikasi berisi pesan "Berhasil mengedit peminjaman venue"
Status	*Valid*

Tabel 6.44 Pengujian Validasi Mengedit Peminjaman Venue Alternatif 2

Kode Kebutuhan	AME_F_1500
Nama Kasus Uji	Mengedit Peminjaman Venue Alternatif 2
Prosedur	1. Aktor mengakses halaman edit peminjaman venue 2. Aktor mengisi *field* nama venue, tanggal peminjaman, waktu mulai dan waktu selesai yang berbenturan dengan peminjaman venue yang lain.
Expected Result	Sistem menampilkan notifikasi berisi pesan error "*Event* akan digunakan oleh Event pada tanggal YYYY-MM-DD"
Result	Sistem menampilkan notifikasi berisi pesan error "*Event* akan digunakan oleh Event pada tanggal YYYY-MM-DD"
Status	*Valid*

Tabel 6.45 Pengujian Validasi Membatalkan Persetujuan *Event*

Kode Kebutuhan	AME_F_1600
Nama Kasus Uji	Membatalkan Persetujuan *Event*
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor menekan *button* batalkan persetujuan
Expected Result	Sistem membatalkan persetujuan event
Result	Sistem membatalkan persetujuan event
Status	*Valid*

Tabel 6.46 Pengujian Validasi Mengedit Persetujuan Event

Kode Kebutuhan	AME_F_1700
Nama Kasus Uji	Mengedit Persetujuan Event
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor mengklik tombol edit persetujuan 3. Aktor mengisikan data persetujuan yang diedit dan

	menekan tombol simpan perubahan
Expected Result	Sistem berhasil mengedit data persetujuan di database, menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan event"
Result	Sistem berhasil mengupdate data persetujuan di database, menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan event"
Status	*Valid*

Tabel 6.47 Pengujian Validasi Mengedit Persetujuan *Event* Alternatif 1

Kode Kebutuhan	AME_F_1700
Nama Kasus Uji	Mengedit Persetujuan *Event* Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor mengklik tombol edit persetujuan 3. Aktor mengisi *field* di form edit persetujuan event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.48 Pengujian Validasi Mengedit Data Pencairan Dana

Kode Kebutuhan	AME_F_1800
Nama Kasus Uji	Mengedit Data Pencairan Dana
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengisi *field* NIP/NIM pengambil dana, nama pengambil dana, tanggal pengambilan dana dan catatan serta menekan tombol simpan perubahan.
Expected Result	Sistem berhasil menyimpan perubahan data pencairan dana yang dilakukan aktor, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit data pencairan dana"
Result	Sistem berhasil menyimpan perubahan data pencairan dana yang dilakukan aktor, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil

	mengedit data pencairan dana
Status	*Valid*

Tabel 6.49 Pengujian Validasi Mengedit Data Pencairan Dana Alternatif 1

Kode Kebutuhan	AME_F_1800
Nama Kasus Uji	Mengedit Data Pencairan Dana Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengisi field di form edit data pencairan dana dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*
Result	Sistem menampilkan notifikasi berisi pesan *error*
Status	*Valid*

Tabel 6.50 Pengujian Validasi Mengedit Persetujuan Event dari Approver Dana

Kode Kebutuhan	AME_F_1900
Nama Kasus Uji	Mengedit Persetujuan Event dari Approver Dana
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengisi field jumlah dana yang disetujui dan catatan kemudian menekan tombol simpan perubahan
Expected Result	Sistem berhasil menyimpan perubahan data persetujuan event ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan *event*"
Result	Sistem berhasil menyimpan perubahan data persetujuan event ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit persetujuan *event*"
Status	*Valid*

Tabel 6.51 Pengujian Validasi Mengedit Persetujuan Event dari Approver Dana Alternatif 1

Kode Kebutuhan	AME_F_1900
Nama Kasus Uji	Mengedit Persetujuan Event dari Approver Dana Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengisi field di form edit persetujuan event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*
Result	Sistem menampilkan notifikasi berisi pesan *error*
Status	*Valid*

Tabel 6.52 Pengujian Validasi Melihat Proposal Event

Kode Kebutuhan	AME_F_2000
Nama Kasus Uji	Melihat Proposal Event
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor menekan link proposal *event*
Expected Result	Sistem menampilkan proposal event di *tab browser* yang lain
Result	Sistem menampilkan proposal event di *tab browser* yang lain
Status	Valid

Tabel 6.53 Pengujian Validasi Melihat Halaman Notifikasi *Event*

Kode Kebutuhan	AME_F_2100
Nama Kasus Uji	Melihat Halaman Notifikasi *Event*
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor menekan tab notifikasi *event*
Expected Result	Sistem menampilkan halaman notifikasi *event*
Result	Sistem menampilkan halaman notifikasi *event*
Status	*Valid*

Tabel 6.54 Pengujian Validasi Melihat Halaman Notifikasi *Event* Alternatif 1

Kode Kebutuhan	AME_F_2100

Nama Kasus Uji	Melihat Halaman Notifikasi Alternatif 1
Prosedur	1. Jika tidak ada notifikasi dan aktor mengakses halaman notifikasi event
Expected Result	Sistem menampilkan notifikasi berisi pesan "Tidak ada notifikasi"
Result	Sistem menampilkan notifikasi berisi pesan "Tidak ada notifikasi"
Status	*Valid*

Tabel 6.55 Pengujian Validasi Menerima Jumlah Dana yang Disetujui

Kode Kebutuhan	AME_F_2200
Nama Kasus Uji	Menerima Jumlah Dana yang Disetujui
Prosedur	1. Aktor mengakses halaman notifikasi event 2. Aktor menekan tombol setujui dana
Expected Result	Sistem berhasil menyimpan persetujuan dana
Result	Sistem berhasil menyimpan persetujuan dana
Status	*Valid*

Tabel 6.56 Pengujian Validasi Menolak Jumlah Dana yang Disetujui

Kode Kebutuhan	AME_F_2300
Nama Kasus Uji	Menolak Jumlah Dana yang Disetujui
Prosedur	1. Aktor mengakses halaman notifikasi event 2. Aktor menekan tombol tolak dana
Expected Result	Sistem berhasil menyimpan tolak dana
Result	Sistem berhasil menyimpan tolak dana
Status	*Valid*

Tabel 6.57 Pengujian Validasi Merevisi Proposal Event

Kode Kebutuhan	AME_F_2400
Nama Kasus Uji	Merevisi Proposal Event
Prosedur	1. Aktor mengakses halaman notifikasi event 2. Aktor mengisi *field file* proposal dan menekan tombol simpan perubahan

Expected Result	Sistem berhasil mengupload file proposal dan mengupdate nama proposal di database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil merevisi proposal"
Result	Sistem berhasil mengupload file proposal dan mengupdate nama proposal di database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil merevisi proposal"
Status	*Valid*

Tabel 6.58 Pengujian Validasi Merevisi Proposal Event Alternatif 1

Kode Kebutuhan	AME_F_2400
Nama Kasus Uji	Merevisi Proposal Event Alternatif 1
Prosedur	1. Aktor mengakses halaman notifikasi event 2. Aktor mengosongkan *field file* proposal kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan "*This field is required*"
Result	Sistem menampilkan notifikasi berisi pesan "*This field is required*"
Status	*Valid*

Tabel 6.59 Pengujian Validasi Melihat Kalender Event

Kode Kebutuhan	AME_F_2500
Nama Kasus Uji	Melihat Kalender Event
Prosedur	1. Aktor mengakses halaman pengajuan *event* 2. Aktor memilih *tab* kalender *event*
Expected Result	Sistem menampilkan kalender *event*
Result	Sistem menampilkan kalender *event*
Status	*Valid*

Tabel 6.60 Pengujian Validasi Melihat List Event dari TU Kemahasiswaan

Kode Kebutuhan	AME_F_2600
Nama Kasus Uji	Melihat *List Event* dari TU Kemahasiswaan
Prosedur	1. Aktor mengakses halaman *list event*

Expected Result	Sistem menampilkan seluruh *event* yang diajukan mahasiswa dan *event* yang pernah diajukan oleh TU Kemahasiswaan
Result	Sistem menampilkan seluruh *event* yang diajukan mahasiswa dan *event* yang pernah diajukan oleh TU Kemahasiswaan
Status	*Valid*

Tabel 6.61 Pengujian Validasi Melihat List Event dari TU Kemahasiswaan Alternatif 1

Kode Kebutuhan	AME_F_2600
Nama Kasus Uji	Melihat List Event dari TU Kemahasiswaan Alternatif 1
Prosedur	1. Aktor belum pernah mengajukan event dan belum ada organisasi mahasiswa yang pernah mengajukan event 2. Aktor mengakses halaman *list event*
Expected Result	Sistem akan menampilkan pesan "*No data available in table*"
Result	Sistem akan menampilkan pesan "*No data available in table*"
Status	*Valid*

Tabel 6.62 Pengujian Validasi Memasukkan Event ke Google Calendar

Kode Kebutuhan	AME_F_2700
Nama Kasus Uji	Memasukkan Event ke Google Calendar
Prosedur	1. Aktor menyetujui peminjaman *venue*
Expected Result	Sistem menyimpan data peminjaman *venue* ke Google Calendar
Result	Sistem menyimpan data peminjaman *venue* ke Google Calendar
Status	*Valid*

Tabel 6.63 Pengujian Validasi Melihat List *Event* dari Approver Non Dekanat

Kode Kebutuhan	AME_F_2800
Nama Kasus Uji	Melihat List Event dari Approver Non Dekanat
Prosedur	1. Aktor mengakses halaman *list* event
Expected Result	Sistem menampilkan event yang pernah disetujui aktor dan event yang pernah diajukan aktor

Result	Sistem menampilkan event yang pernah disetujui aktor dan event yang pernah diajukan aktor
Status	*Valid*

Tabel 6.64 Pengujian Validasi Melihat List Event dari Approver Non Dekanat Alternatif 1

Kode Kebutuhan	AME_F_2800
Nama Kasus Uji	Melihat List Event dari Approver Non Dekanat Alternatif 1
Prosedur	1. Aktor belum pernah mengajukan event dan belum pernah menyetujui event 2. Aktor mengakses halaman *list event*
Expected Result	Sistem akan menampilkan pesan "*No data available in table*"
Result	Sistem akan menampilkan pesan "*No data available in table*"
Status	*Valid*

Tabel 6.65 Pengujian Mengedit Proposal

Kode Kebutuhan	AME_F_2900
Nama Kasus Uji	Mengedit Proposal
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor menekan tombol edit 3. Aktor mengisi *field* file proposal dan menekan tombol simpan perubahan
Expected Result	Sistem berhasil mengupload file proposal dan mengubah nama file proposal di dalam database, sistem menampilkan halaman *list event* dan menampilkan notifikasi pesan "Berhasil mengedit proposal *event*"
Result	Sistem berhasil mengupload file proposal dan mengubah nama file proposal di dalam database, sistem menampilkan halaman *list event* dan menampilkan notifikasi pesan "Berhasil mengedit proposal *event*"
Status	*Valid*

Tabel 6.66 Pengujian Mengedit Proposal Alternatif 1

Kode Kebutuhan	AME_F_2900
Nama Kasus Uji	Mengedit Proposal Alternatif 1

Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengosongkan field file proposal dan menekan tombol simpan perubahan
Expected Result	Sistem menampilkan notifikasi berisi pesan error "*This field is required*"
Result	Sistem menampilkan notifikasi berisi pesan error "*This field is required*"
Status	*Valid*

Tabel 6.67 Pengujian Validasi Membatalkan Penolakan Event

Kode Kebutuhan	AME_F_3000
Nama Kasus Uji	Membatalkan Penolakan Event
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor mengklik button batalkan penolakan
Expected Result	Sistem menyimpan perubahan ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil membatalkan penolakan event"
Result	Sistem menyimpan perubahan ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil membatalkan penolakan event"
Status	*Valid*

Tabel 6.68 Pengujian Validasi Menyetujui Event

Kode Kebutuhan	AME_F_3100
Nama Kasus Uji	Menyetujui Event
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi form menyetujui event yang berisi field catatan dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan *event,* menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Result	Sistem menyimpan data persetujuan *event,* menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Status	*Valid*

Tabel 6.69 Pengujian Validasi Membatalkan Pengajuan Event

Kode Kebutuhan	AME_F_3200
Nama Kasus Uji	Membatalkan pengajuan event
Prosedur	1. Aktor mengakses halaman notifikasi event 2. Aktor mengklik *button* batalkan pengajuan *event*
Expected Result	Sistem menghapus data pengajuan *event* dari *database*
Result	Sistem menghapus data pengajuan *event* dari *database*
Status	*Valid*

Tabel 6.70 Pengujian Validasi Menyetujui Event dari WD2 dan WD3

Kode Kebutuhan	AME_F_3300
Nama Kasus Uji	Menyetujui Event dari WD2 dan WD3
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi form persetujuan event yang terdiri dari field catatan dan dana yang disetujui dan *checkbox* perizinan peminjaman *venue* dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Status	*Valid*

Tabel 6.71 Pengujian Validasi Menyetujui Event dari WD2 dan WD3 Alternatif 1

Kode Kebutuhan	AME_F_3300
Nama Kasus Uji	Menyetujui Event dari WD2 dan WD3 Alternatif 1
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi field di form menetujui event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol setujui.
Expected Result	Sistem tidak menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event

	dan menampilkan notifikasi berisi pesan error
Result	Sistem tidak menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.72 Pengujian Validasi Menyetujui Event dari WD2 dan WD3 Alternatif 2

Kode Kebutuhan	AME_F_3300
Nama Kasus Uji	Menyetujui Event dari WD2 dan WD3 Alternatif 2
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi *form* persetujuan *event* yang terdiri dari *field* catatan dan *checkbox* perizinan peminjaman *venue* dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Status	*Valid*

Tabel 6.73 Pengujian Validasi Menyetujui *Event* dari WD2 dan WD3 Alternatif 3

Kode Kebutuhan	AME_F_3300
Nama Kasus Uji	Menyetujui Event dari WD2 dan WD3 Alternatif 3
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi *form* persetujuan event yang terdiri dari *field* catatan dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"
Result	Sistem menyimpan data persetujuan event ke *database*, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil menyetujui event"

Status	*Valid*

Tabel 6.74 Pengujian Validasi Menolak Event

Kode Kebutuhan	AME_F_3400
Nama Kasus Uji	Menolak Event
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi form persetujuan event yang terdiri dari *field* catatan dan menekan tombol tolak
Expected Result	Sistem berhasil menyimpan data penolakan event
Result	Sistem berhasil menyimpan data penolakan event
Status	*Valid*

Tabel 6.75 Pengujian Validasi Menolak Event Alternatif 1

Kode Kebutuhan	AME_F_3400
Nama Kasus Uji	Menolak Event Alternatif 1
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi field di form tolak event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*
Result	Sistem menampilkan notifikasi berisi pesan *error*
Status	*Valid*

Tabel 6.76 Pengujian Validasi Memasukkan Data Pencairan Dana

Kode Kebutuhan	AME_F_3500
Nama Kasus Uji	Memasukkan Data Pencairan Dana
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi *form* pencairan dana yang terdiri dari *checkbox* verifikasi LPJ, nama pengambil dana, NIM/NIP pengambil dana dan menekan tombol setujui
Expected Result	Sistem menyimpan data pencairan dana ke database, sistem menampilkan halaman persetujuan event dan menampilkan notifikasi berisi pesan “Berhasil memasukkan data pencairan

	dana”
Result	Sistem menyimpan data pencairan dana ke database, sistem menampilkan halaman persetujuan event dan menampilkan notifikasi berisi pesan “Berhasil memasukkan data pencairan dana”
Status	Valid

Tabel 6.77 Pengujian Validasi Memasukkan Data Pencairan Dana Alternatif 1

Kode Kebutuhan	AME_F_3500
Nama Kasus Uji	Memasukkan Data Pencairan Dana Alternatif 1
Prosedur	1. Aktor mengakses halaman persetujuan 2. Aktor mengisi field di form data pencairan dana dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*
Result	Sistem menampilkan notifikasi berisi pesan *error*
Status	Valid

Tabel 6.78 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Umum Perkap

Kode Kebutuhan	AME_F_3600
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Umum Perkap
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field catatan dan menekan tombol setujui untuk event tidak rutin
Expected Result	Sistem berhasil menyimpan data peminjaman venue untuk event tidak rutin ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Berhasil menyetujui event”
Result	Sistem berhasil menyimpan data peminjaman venue untuk event tidak rutin ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Berhasil menyetujui event”
Status	*Valid*

Tabel 6.79 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Umum Perkap Alternatif 1

Kode Kebutuhan	AME_F_3600
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Umum Perkap Alternatif 1
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field di form persetujuan peminjaman venue dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.80 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Umum Perkap Alternatif 2

Kode Kebutuhan	AME_F_3600
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Umum Perkap Alternatif 2
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field catatan dan menekan tombol setujui untuk event rutin
Expected Result	Sistem berhasil menyimpan data peminjaman venue untuk event rutin ke database dan Google Calendar, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Berhasil menyetujui event”
Result	Sistem berhasil menyimpan data peminjaman venue untuk event rutin ke database dan Google Calendar, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan “Berhasil menyetujui event”
Status	*Valid*

Tabel 6.81 Pengujian Validasi Memasukkan Peminjaman Venue

Kode Kebutuhan	AME_F_3700
Nama Kasus Uji	Memasukkan Peminjaman *Venue*

Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab peminjaman venue 3. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, nama lengkap peminjam, NIP/NIM peminjam, email, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol simpan.
Expected Result	Sistem berhasil menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Result	Sistem berhasil menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Status	*Valid*

Tabel 6.82 Pengujian Validasi Memasukkan Peminjaman Venue Alternatif 1

Kode Kebutuhan	AME_F_3700
Nama Kasus Uji	Memasukkan Peminjaman Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab peminjaman venue 3. Aktor mengisi field di form memasukkan peminjaman venue dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.83 Pengujian Validasi Memasukkan Peminjaman Venue Alternatif 2

Kode Kebutuhan	AME_F_3700
Nama Kasus Uji	Memasukkan Peminjaman Venue Alternatif 2
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab peminjaman venue 3. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, nama lengkap peminjam, NIP/NIM

	peminjam, email, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol simpan. 4. Venue yang dipinjam bukan merupakan ruang kelas
Expected Result	Sistem berhasil menyimpan data peminjaman venue ke database dan Google Calendar, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Result	Sistem berhasil menyimpan data peminjaman venue ke database dan Google Calendar, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Status	*Valid*

Tabel 6.84 Pengujian Validasi Memasukkan Peminjaman Venue Alternatif 3

Kode Kebutuhan	AME_F_3700
Nama Kasus Uji	Memasukkan Peminjaman Venue Alternatif 3
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab peminjaman venue 3. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, nama lengkap peminjam, NIP/NIM peminjam, email, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol simpan.
Expected Result	Sistem berhasil menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Result	Sistem berhasil menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Berhasil memasukkan data peminjaman *venue*"
Status	*Valid*

Tabel 6.85 Pengujian Validasi Memasukkan Peminjaman Venue Alternatif 4

Kode Kebutuhan	AME_F_3700

Nama Kasus Uji	Memasukkan Peminjaman Venue Alternatif 4
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab peminjaman venue 3. Aktor mengisi *field* penyelenggara, judul kegiatan, deskripsi *event*, nama lengkap peminjam, NIP/NIM peminjam, email, jumlah personel yang terlibat, *venue* yang dipinjam, tanggal peminjaman, waktu mulai, waktu selesai, lain-lain dan checkbox S&K serta menekan tombol simpan. 4. Data peminjaman venue yang diinputkan aktor bentrok dengan peminjaman venue event lain
Expected Result	Sistem tidak menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Peminjaman venue bentrok dengan: Data peminjaman venue yang bentrok"
Result	Sistem tidak menyimpan data peminjaman venue ke database, sistem menampilkan halaman pengajuan event dan menampilkan notifikasi berisi pesan "Peminjaman venue bentrok dengan: Data peminjaman venue yang bentrok"
Status	*Valid*

Tabel 6.86 Pengujian Validasi Melihat Seluruh List Event

Kode Kebutuhan	AME_F_3800
Nama Kasus Uji	Melihat Seluruh List Event
Prosedur	1. Aktor mengakses halaman list *event*
Expected Result	Sistem menampilkan seluruh *list event* yang ada di sistem
Result	Sistem menampilkan seluruh *list event* yang ada di sistem
Status	*Valid*

Tabel 6.87 Pengujian Validasi Melihat Seluruh List Event Alternatif 1

Kode Kebutuhan	AME_F_3800
Nama Kasus Uji	Melihat Seluruh List Event Alternatif 1
Prosedur	1. Belum ada event yang diajukan 2. Aktor mengakses halaman *list event*
Expected Result	Sistem menampilkan pesan "*No available data in table*"
Result	Sistem menampilkan pesan "*No available data in table*"

Status	*Valid*

Tabel 6.5 Pengujian Validasi Menghapus Event di Google Calendar

Kode Kebutuhan	AME_F_3900
Nama Kasus Uji	Menghapus Event di Google Calendar
Prosedur	1. Aktor menghapus peminjaman venue dari data peminjaman venue yang telah dimasukkan ke Google Calendar
Expected Result	Sistem menghapus data peminjaman venue di Google Calendar
Result	Sistem menghapus data peminjaman venue di Google Calendar
Status	*Valid*

Tabel 6.88 Pengujian Validasi Melihat List Venue

Kode Kebutuhan	AME_F_4000
Nama Kasus Uji	Melihat List Venue
Prosedur	1. Aktor mengakses halaman *list venue*
Expected Result	Sistem menampilkan seluruh *list venue*
Result	Sistem menampilkan seluruh *list venue*
Status	*Valid*

Tabel 6.89 Pengujian Validasi Melihat List Venue Alternatif 1

Kode Kebutuhan	AME_F_4000
Nama Kasus Uji	Melihat List Venue Alternatif 1
Prosedur	1. Belum ada venue di sistem 2. Aktor mengakses halaman list venue
Expected Result	Sistem menampilkan pesan "*No available data in table*"
Result	Sistem menampilkan pesan "*No available data in table*"
Status	*Valid*

Tabel 6.90 Pengujian Validasi Memasukkan Venue Baru

Kode Kebutuhan	AME_F_4100

Nama Kasus Uji	Memasukkan Venue Baru
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan modal tambah venue 3. Aktor mengisi form tambah venue yang terdiri dari field nama venue dan keterangan serta menekan tombol simpan
Expected Result	Sistem menyimpan data venue baru ke database
Result	Sistem menyimpan data venue baru ke database
Status	*Valid*

Tabel 6.91 Pengujian Validasi Memasukkan Venue Baru Alternatif 1

Kode Kebutuhan	AME_F_4100
Nama Kasus Uji	Memasukkan Venue Baru Alternatif 1
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan modal tambah venue 3. Aktor mengisi form tambah venue yang terdiri dari field nama venue dan keterangan serta menekan tombol simpan 4. Aktor menginputkan field nama venue yang telah ada di *database*
Expected Result	Sistem menyimpan notifikasi berisi pesan "Nama *venue* telah ada di dalam database"
Result	Sistem menyimpan notifikasi berisi pesan "Nama *venue* telah ada di dalam database"
Status	*Valid*

Tabel 6.92 Pengujian Validasi Memasukkan Venue Baru Alternatif 2

Kode Kebutuhan	AME_F_4100
Nama Kasus Uji	Memasukkan Venue Baru Alternatif 2
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan modal tambah venue 3. Aktor mengisi field di form masukkan venue baru dengan nilai yang tidak *valid* (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menyimpan notifikasi berisi pesan error
Result	Sistem menyimpan notifikasi berisi pesan error

Status	*Valid*

Tabel 6.93 Pengujian Validasi Memasukkan Venue Menggunakan CSV

Kode Kebutuhan	AME_F_4200
Nama Kasus Uji	Memasukkan Venue Menggunakan CSV
Prosedur	1. Aktor mengakses halaman *list venue* 2. Aktor menekan modal tambah *venue* menggunakan CSV 3. Aktor mengisi field file CSV dan menekan tombol *import*
Expected Result	Sistem berhasil menyimpan seluruh list venue yang ada di dalam file CSV, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Berhasil mengimport venue menggunakan CSV"
Result	Sistem berhasil menyimpan seluruh list venue yang ada di dalam file CSV, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Berhasil mengimport venue menggunakan CSV"
Status	*Valid*

Tabel 6.94 Pengujian Validasi Memasukkan Venue Menggunakan CSV Alternatif 1

Kode Kebutuhan	AME_F_4200
Nama Kasus Uji	Memasukkan Venue Menggunakan CSV Alternatif 1
Prosedur	1. Aktor mengakses halaman *list venue* 2. Aktor menekan modal tambah venue menggunakan CSV 3. Aktor memilih file CSV yang tidak sesuai format di database venue dan menekan tombol *import*
Expected Result	Sistem menampilkan notifikasi berisi pesan *error*
Result	Sistem menampilkan notifikasi berisi pesan *error*
Status	*Valid*

Tabel 6.95 Pengujian Validasi Memasukkan Venue Menggunakan CSV Alternatif 2

Kode Kebutuhan	AME_F_4200

Nama Kasus Uji	Memasukkan Venue Menggunakan CSV Alternatif 2
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan modal tambah venue menggunakan CSV 3. Aktor mengosongkan field file CSV dan menekan tombol import
Expected Result	Sistem menampilkan notifikasi berisi pesan error "*This field is required*"
Result	Sistem menampilkan notifikasi berisi pesan error "*This field is required*"
Status	*Valid*

Tabel 6.96 Pengujian Validasi Menghapus Venue

Kode Kebutuhan	AME_F_4300
Nama Kasus Uji	Menghapus Venue
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan tombol hapus dan menekan tombol konfirmasi penghapusan
Expected Result	Sistem menghapus venue dari database
Result	Sistem menghapus venue dari database
Status	*Valid*

Tabel 6.97 Pengujian Validasi Menghapus Venue Alternatif 1

Kode Kebutuhan	AME_F_4300
Nama Kasus Uji	Menghapus Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan tombol hapus dan menekan tombol batal di alert konfirmasi
Expected Result	Sistem membatalkan penghapusan venue dari database
Result	Sistem membatalkan penghapusan venue dari database
Status	*Valid*

Tabel 6.98 Pengujian Validasi Mengedit Venue

Kode Kebutuhan	AME_F_4400

Nama Kasus Uji	Mengedit Venue
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan tombol edit 3. Aktor mengisi field nama venue atau keterangan dan menekan tombol simpan perubahan.
Expected Result	Sistem menyimpan data perubahan venue kedalam database, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Berhasil mengedit venue"
Result	Sistem menyimpan data perubahan venue kedalam database, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Berhasil mengedit venue"
Status	*Valid*

Tabel 6.99 Pengujian Validasi Mengedit Venue Alternatif 1

Kode Kebutuhan	AME_F_4400
Nama Kasus Uji	Mengedit Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan tombol edit 3. Aktor mengubah nama venue dengan nama venue yang telah ada di database dan menekan tombol simpan perubahan
Expected Result	Sistem tidak menyimpan data perubahan venue kedalam database, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Nama venue telah ada di database"
Result	Sistem tidak menyimpan data perubahan venue kedalam database, sistem menampilkan halaman list venue dan menampilkan notifikasi berisi pesan "Nama venue telah ada di database"
Status	*Valid*

Tabel 6.100 Pengujian Validasi Mengedit Venue Alternatif 2

Kode Kebutuhan	AME_F_4400
Nama Kasus Uji	Mengedit Venue Alternatif 2
Prosedur	1. Aktor mengakses halaman list venue 2. Aktor menekan tombol edit

	3. Aktor mengisi field di form edit venue dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.101 Pengujian Validasi Mencari Venue

Kode Kebutuhan	AME_F_4500
Nama Kasus Uji	Mencari Venue
Prosedur	1. Aktor mengakses halaman *list event* 2. Aktor mengisi field pencarian venue
Expected Result	Sistem menampilkan *venue* yang dicari aktor
Result	Sistem menampilkan *venue* yang dicari aktor
Status	*Valid*

Tabel 6.102 Pengujian Validasi Mencari Venue Alternatif 1

Kode Kebutuhan	AME_F_4500
Nama Kasus Uji	Mencari Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor mengisi mencari *venue* yang tidak ada di *database*
Expected Result	Sistem menampilkan notifikasi "*No matching records found*"
Result	Sistem menampilkan notifikasi "*No matching records found*"
Status	*Valid*

Tabel 6.103 Pengujian Validasi Mengedit Event di Google Calendar

Kode Kebutuhan	AME_F_4600
Nama Kasus Uji	Mengedit Event di Google Calendar
Prosedur	1. Aktor mengedit peminjaman venue dari data peminjaman venue yang telah dimasukkan ke Google Calendar
Expected Result	Sistem mengedit data peminjaman venue di Google Calendar

Result	Sistem mengedit data peminjaman venue di Google Calendar
Status	Valid

Tabel 6.104 Pengujian Validasi Melihat List Peminjaman Venue dari TU Akademik

Kode Kebutuhan	AME_F_4700
Nama Kasus Uji	Melihat List Peminjaman Venue dari TU Akademik
Prosedur	1. Aktor mengakses halaman list peminjaman *venue*
Expected Result	Sistem menampilkan peminjaman venue ruang kelas
Result	Sistem menampilkan peminjaman venue ruang kelas
Status	*Valid*

Tabel 6.105 Pengujian Validasi Melihat List Peminjaman Venue dari TU Akademik Alternatif 1

Kode Kebutuhan	AME_F_4700
Nama Kasus Uji	Melihat List Peminjaman Venue dari TU Akademik Alternatif 1
Prosedur	1. Belum ada peminjaman venue 2. Aktor mengakses halaman list peminjaman venue
Expected Result	Sistem menampilkan notifikasi yang berisi pesan "*No data available in table*"
Result	Sistem menampilkan notifikasi yang berisi pesan "*No data available in table*"
Status	*Valid*

Tabel 6.106 Pengujian Validasi Membuat Akun Baru

Kode Kebutuhan	AME_F_4800
Nama Kasus Uji	Membuat Akun Baru
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol tambah akun 3. Aktor mengisi field NIP/NIM, nama unit/organisasi, nama ketua unit/organisasi, email, no HP, role dan keterangan kemudian menekan tombol simpan
Expected Result	Sistem berhasil menyimpan data akun baru ke database, sistem mengirimkan email ke email pembuat akun, sistem

	menampilkan halaman list akun dan menampilkan notifikasi berisi pesan “Berhasil menambahkan akun”
Result	Sistem berhasil menyimpan data akun baru ke database, sistem mengirimkan email ke email pembuat akun, sistem menampilkan halaman list akun dan menampilkan notifikasi berisi pesan “Berhasil menambahkan akun”
Status	*Valid*

Tabel 6.107 Pengujian Validasi Membuat Akun Baru Alternatif 1

Kode Kebutuhan	AME_F_4800
Nama Kasus Uji	Membuat Akun Baru Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol tambah akun 3. Aktor mengisi field di form tambah akun dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.108 Pengujian Validasi Membuat Akun Baru Alternatif 2

Kode Kebutuhan	AME_F_4800
Nama Kasus Uji	Membuat Akun Baru Alternatif 2
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol tambah akun 3. Aktor mengisi *field* NIP/NIM yang telah ada di *database*, nama unit/organisasi, nama ketua unit/organisasi, email, no HP, role dan keterangan kemudian menekan tombol simpan
Expected Result	Sistem menampilkan notifikasi berisi pesan notifikasi error “NIP/NIM telah digunakan”
Result	Sistem menampilkan notifikasi berisi pesan notifikasi error “NIP/NIM telah digunakan”
Status	*Valid*

Tabel 6.109 Pengujian Validasi Melihat List Akun

Kode Kebutuhan	AME_F_4900
Nama Kasus Uji	Melihat List Akun
Prosedur	1. Aktor mengakses halaman list akun
Expected Result	Sistem menampilkan seluruh list akun
Result	Sistem menampilkan seluruh list akun
Status	*Valid*

Tabel 6.110 Pengujian Validasi Mencari Akun

Kode Kebutuhan	AME_F_5000
Nama Kasus Uji	Mencari Akun
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor mengisi *field* pencarian akun
Expected Result	Sistem menampilkan akun yang dicari aktor
Result	Sistem menampilkan akun yang dicari aktor
Status	*Valid*

Tabel 6.111 Pengujian Validasi Mencari Akun Alternatif 1

Kode Kebutuhan	AME_F_5000
Nama Kasus Uji	Mencari Akun Alternatif 1
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor mencari akun yang tidak ada di sistem
Expected Result	Sistem menampilkan pesan "*No data available in table*"
Result	Sistem menampilkan pesan "*No data available in table*"
Status	*Valid*

Tabel 6.112 Pengujian Validasi Mengedit Akun

Kode Kebutuhan	AME_F_5100
Nama Kasus Uji	Mengedit Akun
Prosedur	1. Aktor mengakses halaman edit akun 2. Aktor mengisi field NIP/NIM, nama unit/organisasi, nama ketua unit/organisasi, email, no HP, role atau keterangan kemudian menekan tombol simpan

Expected Result	Sistem menyimpan perubahan data akun ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit akun"
Result	Sistem menyimpan perubahan data akun ke database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mengedit akun"
Status	*Valid*

Tabel 6.113 Pengujian Validasi Mengedit Akun Alternatif 1

Kode Kebutuhan	AME_F_5100
Nama Kasus Uji	Mengedit Akun Alternatif 1
Prosedur	1. Aktor mengakses halaman edit akun 2. Aktor mengisi field di form edit akun dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.114 Pengujian Validasi Mengedit Akun Alternatif 2

Kode Kebutuhan	AME_F_5100
Nama Kasus Uji	Mengedit Akun Alternatif 2
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor menekan tombol edit 3. Aktor mengubah field NIP/NIM dengan NIP/NIM yang telah ada di database kemudian menekan tombol simpan perubahan
Expected Result	Sistem menampilkan notifikasi berisi pesan error "NIP/NIM telah digunakan"
Result	Sistem menampilkan notifikasi berisi pesan error "NIP/NIM telah digunakan"
Status	*Valid*

Tabel 6.115 Pengujian Validasi Menghapus Akun

Kode Kebutuhan	AME_F_5200

Nama Kasus Uji	Menghapus Akun
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol hapus dan konfirmasi penghapusan
Expected Result	Sistem menghapus data akun dari database, sistem menampilkan halaman list akun dan menampilkan notifikasi berisi pesan "Berhasil menghapus akun"
Result	Sistem menghapus data akun dari database, sistem menampilkan halaman list akun dan menampilkan notifikasi berisi pesan "Berhasil menghapus akun"
Status	*Valid*

Tabel 6.116 Pengujian Validasi Menghapus Akun Alternatif 1

Kode Kebutuhan	AME_F_5200
Nama Kasus Uji	Menghapus Akun Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol hapus dan batal di alert konfirmasi penghapusan akun
Expected Result	Sistem membatalkan penghapusan akun
Result	Sistem membatalkan penghapusan akun
Status	*Valid*

Tabel 6.117 Pengujian Validasi Mereset Password

Kode Kebutuhan	AME_F_5300
Nama Kasus Uji	Mereset Password
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor menekan button reset dan konfirmasi reset
Expected Result	Sistem mengupdate password aktor di database, mengirimkan email ke email aktor yang berisi password baru, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mereset password akun"
Result	Sistem mengupdate password aktor di database, mengirimkan email ke email aktor yang berisi password baru, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil mereset password akun"
Status	Valid

Tabel 6.118 Pengujian Validasi Mereset Password Alternatif 1

Kode Kebutuhan	AME_F_5300
Nama Kasus Uji	Mereset Password Alternatif 1
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor menekan button reset dan batal di alert konfirmasi konfirmasi
Expected Result	Sistem membatalkan proses peresetan password
Result	Sistem membatalkan proses peresetan password
Status	Valid

Tabel 6.119 Pengujian Validasi Melihat Seluruh List Peminjaman Venue

Kode Kebutuhan	AME_F_5400
Nama Kasus Uji	Melihat Seluruh List Peminjaman Venue
Prosedur	1. Aktor mengakses halaman list peminjaman venue
Expected Result	Sistem menampilkan seluruh list peminjaman venue
Result	Sistem menampilkan seluruh list peminjaman venue
Status	*Valid*

Tabel 6.120 Pengujian Validasi Melihat Seluruh List Peminjaman Venue Alternatif 1

Kode Kebutuhan	AME_F_5400
Nama Kasus Uji	Melihat Seluruh List Peminjaman Venue Alternatif 1
Prosedur	1. Belum ada peminjaman venue 2. Aktor mengakses halaman list peminjaman venue
Expected Result	Sistem menampilkan notifikasi berisi pesan "*No data available in table*"
Result	Sistem menampilkan notifikasi berisi pesan "*No data available in table*"
Status	*Valid*

Tabel 6.121 Pengujian Validasi Menghapus Peminjaman Venue

Kode Kebutuhan	AME_F_5500

Nama Kasus Uji	Menghapus Peminjaman Venue
Prosedur	1. Aktor mengakses halaman list peminjaman venue 2. Aktor menekan tombol hapus dan konfirmasi penghapusan
Expected Result	Sistem menghapus data peminjaman venue dari database, sistem menampilkan halaman list peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menghapus peminjaman venue"
Result	Sistem menghapus data peminjaman venue dari database, sistem menampilkan halaman list peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menghapus peminjaman venue"
Status	*Valid*

Tabel 6.122 Pengujian Validasi Menghapus Peminjaman Venue Alternatif 1

Kode Kebutuhan	AME_F_5500
Nama Kasus Uji	Menghapus Peminjaman Venue Alternatif 1
Prosedur	1. Aktor mengakses halaman list peminjaman venue 2. Aktor menekan tombol hapus dan batal pada alert konfirmasi penghapusan venue
Expected Result	Sistem membatalkan penghapusan peminjaman venue dari database
Result	Sistem membatalkan penghapusan peminjaman venue dari database
Status	*Valid*

Tabel 6.123 Pengujian Validasi Menghapus Peminjaman Venue Alternatif 2

Kode Kebutuhan	AME_F_5500
Nama Kasus Uji	Menghapus Peminjaman Venue Alternatif 2
Prosedur	1. Aktor mengakses halaman list peminjaman venue 2. Aktor menekan tombol hapus dan konfirmasi penghapusan
Expected Result	Sistem menghapus data peminjaman venue dari database, menghapus data peminjaman venue di Google Calendar, sistem menampilkan halaman list peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menghapus peminjaman venue"

Result	Sistem menghapus data peminjaman venue dari database, menghapus data peminjaman venue di Google Calendar, sistem menampilkan halaman list peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menghapus peminjaman venue"
Status	*Valid*

Tabel 6.124 Pengujian Validasi Menghapus Event

Kode Kebutuhan	AME_F_5600
Nama Kasus Uji	Menghapus Event
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol hapus dan konfirmasi penghapusan
Expected Result	Sistem menghapus event dari database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil menghapus event"
Result	Sistem menghapus event dari database, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil menghapus event"
Status	*Valid*

Tabel 6.125 Pengujian Validasi Menghapus Event Alternatif 1

Kode Kebutuhan	AME_F_5600
Nama Kasus Uji	Menghapus Event Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol hapus dan batalkan pada alert konfirmasi penghapusan
Expected Result	Sistem membatalkan penghapusan event dari database
Result	Sistem membatalkan penghapusan event dari database
Status	*Valid*

Tabel 6.126 Pengujian Validasi Menghapus Event Alternatif 2

Kode Kebutuhan	AME_F_5600
Nama Kasus Uji	Menghapus Event Alternatif 2
Prosedur	1. Aktor mengakses halaman list event 2. Aktor memilih peminjaman *venue* yang telah

	dimasukkan ke Google Calendar 3. Aktor menekan tombol hapus dan konfirmasi penghapusan
Expected Result	Sistem menghapus event dari database, sistem menghapus peminjaman venue dari Google Calendar, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil menghapus event"
Result	Sistem menghapus event dari database, sistem menghapus peminjaman venue dari Google Calendar, sistem menampilkan halaman list event dan menampilkan notifikasi berisi pesan "Berhasil menghapus event"
Status	*Valid*

Tabel 6.127 Pengujian Mengedit Data Event

Kode Kebutuhan	AME_F_5700
Nama Kasus Uji	Mengedit Data Event
Prosedur	1. Aktor mengakses halaman edit event 2. Aktor mengisi field judul kegiatan deskripsi event atau jenis event menekan button simpan perubahan
Expected Result	Sistem menyimpan data perubahan event ke database, sistem menampilkan halaman edit event dan menampilkan notifikasi berisi pesan "Berhasil mengedit data event"
Result	Sistem menyimpan data perubahan event ke database, sistem menampilkan halaman edit event dan menampilkan notifikasi berisi pesan "Berhasil mengedit data event"
Status	*Valid*

Tabel 6.128 Pengujian Mengedit Data Event Alternatif 1

Kode Kebutuhan	AME_F_5700
Nama Kasus Uji	Mengedit Data Event Alternatif 1
Prosedur	1. Aktor mengakses halaman list event 2. Aktor menekan tombol edit 3. Aktor mengisi field di form edit data event dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error.

Result	Sistem menampilkan notifikasi berisi pesan error.
Status	*Valid*

Tabel 6.129 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Akademik

Kode Kebutuhan	AME_F_5800
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Akademik
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field catatan dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan venue di database, sistem menampilkan halaman peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menyetujui peminjaman venue".
Result	Sistem menyimpan data persetujuan venue di database, sistem menampilkan halaman peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menyetujui peminjaman venue".
Status	*Valid*

Tabel 6.130 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Akademik Alternatif 1

Kode Kebutuhan	AME_F_5800
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Akademik Alternatif 1
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field catatan dan menekan tombol setujui
Expected Result	Sistem menyimpan data persetujuan venue di database, mengedit data persetujuan event di Google Calendar, sistem menampilkan halaman peminjaman venue dan menampilkan notifikasi berisi pesan "Berhasil menyetujui peminjaman venue".
Result	Sistem menyimpan data persetujuan venue di database, mengedit data persetujuan event di Google Calendar, sistem menampilkan halaman peminjaman venue dan menampilkan

	notifikasi berisi pesan "Berhasil menyetujui peminjaman venue".
Status	*Valid*

Tabel 6.131 Pengujian Validasi Menyetujui Peminjaman Venue dari TU Akademik Alternatif 2

Kode Kebutuhan	AME_F_5800
Nama Kasus Uji	Menyetujui Peminjaman Venue dari TU Akademik Alternatif 2
Prosedur	1. Aktor mengakses halaman pengajuan event 2. Aktor mengklik tab persetujuan event 3. Aktor mengisi field di form persetujuan peminjaman venue dengan nilai yang tidak valid (misalnya ada field yang dikosongkan) kemudian menekan tombol simpan perubahan.
Expected Result	Sistem menampilkan notifikasi berisi pesan error
Result	Sistem menampilkan notifikasi berisi pesan error
Status	*Valid*

Tabel 6.132 Pengujian Validasi Menampilkan Detail Akun

Kode Kebutuhan	AME_F_5900
Nama Kasus Uji	Menampilkan Detail Akun
Prosedur	1. Aktor mengakses halaman list akun 2. Aktor menekan tombol detail akun
Expected Result	Sistem menampilkan data akun dan event yang pernah diajukan melalui akun tersebut.
Result	Sistem menampilkan data akun dan event yang pernah diajukan melalui akun tersebut.
Status	*Valid*

6.3 Pengujian *Browser Compatibility*

Pada pengujian *browser compatibility*, penulis menggunakan metode *automate testing*. Untuk melakukan *automate testing* ini, penulis menggunakan *tool* bernama SortSite. SortSite bekerja dengan cara memeriksa beberapa hal berikut ini:

a. *Tag* HTML yang tidak didukung oleh beberapa *browser*

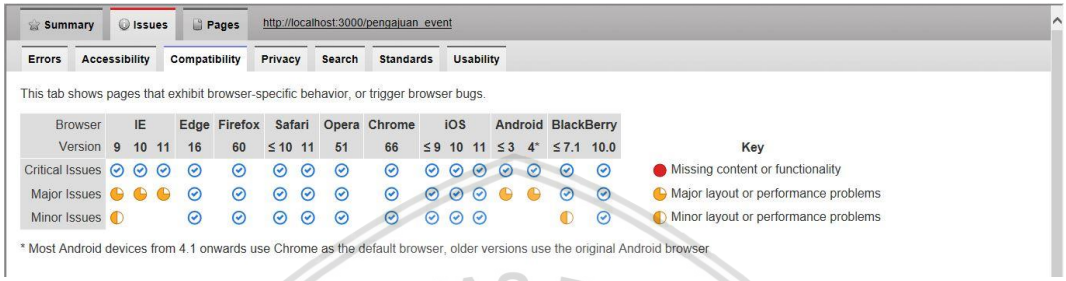
b. Fitur CSS yang tidak didukung oleh beberapa *browser*

c. *Vendor specific* HTML dan Javascript

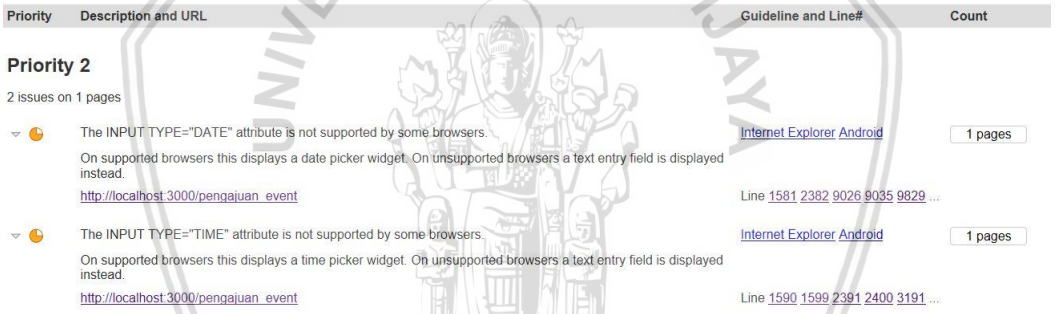
d. Format gambar yang tidak didukung beberapa *browser*

e. Teknologi yang tidak didukung oleh beberapa *browser*

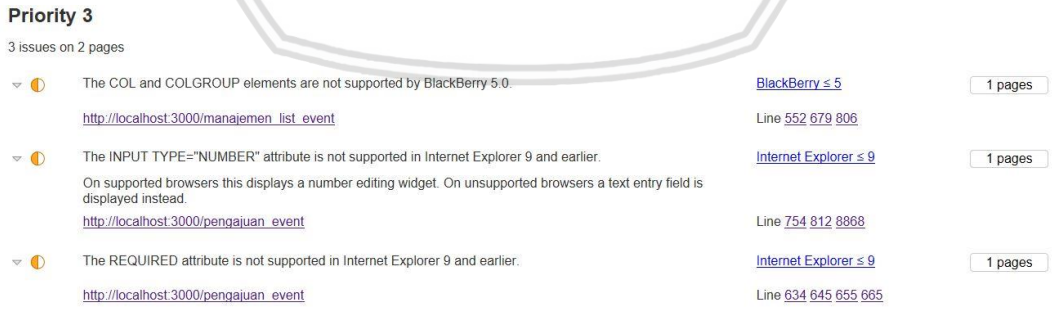
Gambar 6.4 – 6.6 dibawah ini merupakan hasil dari pengujian *browser compatibility* menggunakan *tool* SortSite.



Gambar 6.4 Hasil Pengujian *Browser Compatibility* 1



Gambar 6.5 Hasil Pengujian *Browser Compatibility* 2



Gambar 6.6 Hasil Pengujian *Browser Compatibility* 2

Pada Gambar 6.1 dapat dilihat aplikasi berjalan tanpa kehilangan konten atau fungsionalitas, tidak ada masalah *major* pada *layout* dan tidak ada masalah *minor* pada *layout* di *browser* Edge, Firefox, Safari Opera dan Chrome. Sehingga dapat disimpulkan aplikasi dapat berjalan secara sempurna di *browser* Edge, Firefox, Safari, Opera dan Chrome.

Namun, terdapat masalah *major* dan masalah *minor* yang muncul pada *browser* Internet Explorer. Seperti pada Gambar 6.5 dan Gambar 6.6 masalah *major* yang muncul adalah Internet Explorer tidak men*support input* yang memiliki tipe *date* dan *time*. Sedangkan masalah *minor* yang timbul adalah Internet Explorer tidak men*support input* dengan tipe *number* serta *input* yang memiliki attribut *required*. Sehingga dapat disimpulkan aplikasi tidak dapat berjalan secara baik di *browser* Internet Explorer.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Kesimpulan diambil berdasarkan pada hasil dari tahap analisis kebutuhan, perancangan, implementasi dan pengujian yang sebelumnya telah dilakukan di penelitian ini. Kesimpulan menjawab semua permasalahan yang telah didefinisikan pada rumusan masalah. Sehingga pada penelitian ini dapat ditarik kesimpulan sebagai berikut:

1. Berdasarkan hasil analisis kebutuhan, aplikasi Manajemen *Event* mempunyai 59 kebutuhan fungsional dan 1 kebutuhan non-fungsional serta memiliki 16 aktor yang terdiri dari 15 aktor primer dan 1 aktor sekunder. Selain itu pada tahap analisis kebutuhan telah dibuat diagram *Business Process Modelling* (BPM) yang mengilustrasikan prosedur pengajuan *event* di Fakultas Ilmu Administrasi Universitas Brawijaya. Pada pemodelan kebutuhan terdapat *use case diagram* yang berisi 59 *use case* kemudian setiap *use case* didetailkan di *use case scenario*. Kemudian penulis juga membuat *State Transition Diagram* (STD) yang menggambarkan alur dan status disposisi proposal atau persetujuan *event*.
2. Pada tahap perancangan aplikasi Manajemen *Event* ini, telah dibuat *sequence diagram* dari 3 kebutuhan yang ada di aplikasi Manajemen *Event* sebagai sample. Pada *class diagram* diperoleh 8 kelas yang terdiri dari 4 kelas *controller* dan 3 kelas *model* serta 1 kelas dari aplikasi eksternal yaitu Google Calendar. Pada *entity relationship diagram* diperoleh 5 entitas dimana kelima entitas tersebut dihubungkan dengan 4 relasi. Pada perancangan komponen telah dibuat *pseudocode* dari 3 *method* yang ada di aplikasi ini serta pada perancangan *user interface* penulis membuat *mock-up* dari 4 halaman yang ada di aplikasi ini.
3. Pada tahap implementasi kode program penulis menggunakan Node.js dengan menggunakan bantuan *framework* Express.js dan berdasar pada *pseudocode* yang telah dibuat pada tahap perancangan. Selain itu untuk mengimplementasikan Google Calendar API, penulis menggunakan *package* node-google-calendar. Untuk mengimplementasikan *user interface,* penulis menggunakan *framework* SemanticUI dan jQuery dan berdasar pada perancangan antarmuka yang telah dibuat pada tahap perancangan. Untuk implementasi data, penulis membuat *physical data model.* Pada *physical data model* terdapat 7 entitas.
4. Pada pengujian *unit,* seluruh *independent paths* telah diuji dan seluruhnya *valid*. Hasil pengujian validasi juga menunjukkan nilai 100% *valid***.** Sedangkan pada pengujian *browser compatibility* diperoleh hasil yang menunjukkan bahwa aplikasi dapat dijalankan secara sempurna di peramban Opera, Chrome, Firefox, Edge dan Safari. Namun terdapat *major* dan *minor problems* saat aplikasi dijalankan melalui peramban

Internet Explorer, tetapi seluruh fungsionalitas tetap dapat dijalankan/tidak sampai terjadi *missing functionality*.

7.2 Saran

Saran yang dapat diberikan dari penelitian yang telah dilakukan dan dapat digunakan untuk pengembangan sistem dan penelitian selanjutnya adalah:

1. Melakukan penambahkan fitur untuk menyimpan jadwal kuliah rutin dan jadwal kuliah pengganti. Sehingga sistem selain dapat mengecek ketersediaan *venue* yang bukan merupakan ruang kelas, juga dapat mengecek ketersediaan *venue* yang merupakan ruang kelas.
2. Perlunya pengimplementasian *transaction* pada beberapa proses yang berhubungan dengan manipulasi *database* yang bertujuan untuk menjaga *consistency* data.
3. Saran agar mengganti pengimplementasian *promise* dengan *async await* agar kode lebih *maintainable.*
4. Mengganti penggunaan aplikasi eksternal yaitu Google Calendar dengan mengimplementasikan kalender sendiri. Hal ini bertujuan untuk menurunkan *coupling* dari aplikasi Manajemen *Event*.

DAFTAR PUSTAKA

Alshamrani, A., dan Bahattab, A., 2015. *A Comparison Between Three SDLC Models Waterfall, Spiral Model, and Incremental/Iterative Model*, [e-journal] pp 106-111. Tersedia melalui: IJCSI <https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf> [Diakses 22 September 2017]

Chaniotis, et. al., 2014. *Is Node.js a viable option for building modern web Applications? A performance evaluation study*, [e-journal] pp 1023-1044. Tersedia melalui: Springer <https://link.springer.com/article/10.1007/s00607-014-0394-9> [Diakses 22 September 2017]

Dalbey, J., 2017. *Examples of State Transition Diagrams* [online]. Tersedia melalui: <http://users.csc.calpoly.edu/~jdalbey/SWE/Design/STDexamples.Html> [Diakses 22 September 2017]

Express.js, 2017. *Express.js*, [online]. Tersedia melalui: <https://expressjs.com/> [Diakses 16 September 2017]

Github, 2017. Nodemon, [online]. Tersedia melalui:<https://github.com/remy/nodemon> [Diakses 22 September 2017]

Lei, K., et. al., 2014. *Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js*, [e-journal] pp 661-668. IEEE. Tersedia melalui: IEEE <http://ieeexplore.ieee.org/document/7023652/> [Diakses 22 September 2017]

Mayven, 2017. *5 Popular Node.js Frameworks (And What They Do),* [online]. Tersedia melalui: <https://mayvendev.com/blog/5-popular-node-js-frameworks> [Diakses 22 September 2017]

Meiliana, 2017. *Basis Path Testing: Flow Graph*. Tersedia melalui: <http://socs.binus.ac.id/2016/12/30/basis-path-testing-flow-graph/> [Diakses 22 September 2017]

Noor, A., 2013. *Manajemen Event*. Bandung: Alfabeta

Npm, 2017. *Express-mysql-session,* [online]. Tersedia melalui: <https://www.npmjs.com/package/express-mysql-session> [Diakses 22 September 2017]

Npm, 2017. *Node-google-calendar,* [online]. Tersedia melalui: < https://www.npmjs.com/package/node-google-calendar> [Diakses 22 September 2017]

Ojamaa, A., dan Duuna, K., 2012. *Assessing the Security of Node.js Platform*, [e-journal] pp 348-355. Tersedia melalui: IEEE < http://ieeexplore.ieee.org/document/6470829/>. [Diakses 22 September 2017]

Passport.js, 2017. *Overview,* [online]. Tersedia melalui: < http://www.passportjs.org/docs/downloads/html/> [Diakses 22 September 2017]

Pressman, R. S., Maxim B. R., 2015. *Software Engineering A Practitioner's Approach 8th Edition*, [e-book]. New York: Mc Graw Hill Education. Tersedia melalui: <https://downloadnema.com/wpcontent/uploads/2017/02/Software%20Engineering%20A%20Practitioner%E2%80%99s%20Approach%20eighth%20edition-(www.downloadnema.com).pdf> [Diakses 22 September 2017]

Scotts, D., 2000. *State Transition Diagrams* [online]. Tersedia melalui : <http:// www.cs.unc.edu/~stotts /145/CRC/state.html> [Diakses 7 Januari 2018]

SemanticUI, 2017. *Modal,* [online]. Tersedia melalui: <https://semantic-ui.com/modules/modal.html > [Diakses 22 September 2017]

Sommerville, I., 2011. *Software Engineering Ninth Edition*, [e-book]. London: AddisonWesley. Tersedia melalui: <https://www.pdfdrive.net/softwareengineering-9th-edition-e12180831.html> [Diakses 22 September 2017]